GLOBAL INITIATIVE OF ACADEMIC NETWORKS
(GIAN)

Solving linear systems and computing generalized inverses using recurrent neural networks

9th June to 19th June 2025
IIT Indore.

GIAN
GLOBAL INITIATIVE OF ACADEMIC NETWORKS

Generalized Matrix Inversion: A Machine Learning Approach

Predrag S. Stanimirović
University of Niš,
Faculty of Sciences and Mathematics,
Višegradska 33, 18000 Niš, Serbia

June 13, 2025

Introduction

The Kronecker product of an $m \times n$ matrix A and an $r \times q$ matrix B, is an $mr \times nq$ matrix, denoted by $A \otimes B$ and defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & & & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}.$$

Main properties of the Kronecker product are as follows.

$$\begin{aligned} A \otimes (B + C) &= A \otimes B + A \otimes C; \\ A \otimes (B \otimes C) &= (A \otimes B) \otimes C; \\ \alpha A \otimes \beta B &= \alpha\beta(A \otimes B); \\ (A \otimes B)^{\mathrm{T}} &= A^{\mathrm{T}} \otimes B^{\mathrm{T}}; \\ (A \otimes B)^{-1} &= A^{-1} \otimes B^{-1}; \\ (A \otimes B)(C \otimes D) &= AC \otimes BD; \\ \mathrm{rank}(A \otimes B) &= \mathrm{rank}(A)\mathrm{rank}(B); \\ \mathrm{Tr}(A \otimes B) &= \mathrm{Tr}(A)\mathrm{Tr}(B); \end{aligned}$$

Introduction

The *vectorization* (vec-operator) applied on a matrix A stacks the columns into a vector. If $a_1, \ldots a_n$ are columns of a matrix A, then

$$\mathrm{vec}(A) = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}.$$

The vectorization has the following properties:

$$\begin{aligned} \mathrm{vec}(A+B) &= \mathrm{vec}(A) + \mathrm{vec}(B) \\ \mathrm{vec}(ABC) &= \left(C^{\mathrm{T}} \otimes A\right) \mathrm{vec}(B) \\ \mathrm{vec}(AB) &= (I_p \otimes A)\mathrm{vec}(B) = (B^{\mathrm{T}} \otimes I_m)\mathrm{vec}(A), \end{aligned}$$

where $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{n\times p}$, $C \in \mathbb{C}^{p\times q}$.

Lecture 4:
- Recurrent Neural Networks (RNN),
- Continuous-time RNN, Gradient Neural Networks (GNN),
- GNN dynamics for solving linear matrix equations AXB=C,
- GNN for computing generalized inverses of constant matrices,
- GNN for solving systems of linear equations.

Introduction

There are two types of artificial neural networks (ANNs).
1. A feedforward neural network (FNN) is a type of ANNs where information moves in one direction, from the input layer to the output layer, without any loops or cycles.
2. Recurrent Neural Networks (RNN) is one of the two broad types of ANNs that allow the output from some nodes to affect subsequent input to the same nodes.

A continuous-time recurrent neural network (CTRNN) uses a system of ordinary differential equations to define effects on incoming inputs on a neuron.

We will consider CTRNN dedicated to find zeros of equations or to minimize nonlinear functions.

Two important classes of CTRNNs are known::
Gradient Neural Networks (GNN) and
Zhang Neural Networks (ZNN).

Our main objective is to describe the application of GNN dynamical systems in solving numerical linear algebra problems, mainly in:

solving matrix equations,

solving systems of linear equations,

computing generalized inverses.

GNN dynamics

Main step in defining GNN dynamics.

Step1GNN. Define an appropriate error matrix $E(t)$ on the basis of the matrix equation which is currently being solved.

The error matrix $E(t)$ is defined by replacing the unknown matrix from the considered problem by the time-varying matrix $V(t)$ which will be approximated during the time $t \geq 0$.

For example, the matrix equation $AX = I$ is appropriate for computing the inverse A^{-1}.
In this case, $E(t) = AV(t) - I$.

GNN dynamics

The goal function of a Gradient Neural Network (GNN) model is the scalar function which is defined by the Frobenius norm of $E(t)$:

$$\varepsilon(t) = \frac{\|E(t)\|_F^2}{2}, \quad \|E\|_F = \sqrt{\mathrm{Tr}(E^{\mathrm{T}}E)}.$$

Step2GNN. Apply the dynamic evolution, which is based on direct proportionality between the time derivative $\dot{V}(t)$ and the scaled negative gradient of the goal function $\varepsilon(t)$:

$$\dot{V}(t) = \frac{\mathrm{d}V(t)}{\mathrm{d}t} = -\gamma \frac{\partial \varepsilon(t)}{\partial V}, \quad V(0) = V_0. \tag{1}$$

Here, $V(t)$ is the matrix of activation state variables and $t \in [0, +\infty)$ is the time. Larger values of the scaling parameter γ enable faster convergence.

GNN model

Assume that the discretization of the continuous-time linear GNN model is performed by using the Euler forward-difference rule

$$\dot{V}(t) \approx (V_{k+1} - V_k)/\tau,$$

where τ denotes the sampling time and $V_k = V(t = k\tau)$, $k = 1, 2, \ldots$.

Then the discrete-time analogy of the GNN model is just a gradient descent method for nonlinear optimization:

$$V_{k+1} = V_k - \beta \nabla \varepsilon(V_k), \tag{2}$$

where $\beta = \tau \gamma > 0$ is the step size.
Clearly, smaller sampling time τ can be achieved using larger values γ.

Introduction

The following properties of the matrix derivatives of the matrix trace are frequently exploited.

(a) $\dfrac{\partial}{\partial X}\mathrm{Tr}(X) = I;$

(b) $\dfrac{\partial}{\partial X}\mathrm{Tr}(XA) = \dfrac{\partial}{\partial X}\mathrm{Tr}(AX) = A^{\mathrm{T}};$

(c) $\dfrac{\partial}{\partial X}\mathrm{Tr}(AXB) = A^{\mathrm{T}}B^{\mathrm{T}};$

(d) $\dfrac{\partial}{\partial X}\mathrm{Tr}(AX^{\mathrm{T}}B) = BA;$

(e) $\dfrac{\partial}{\partial X}\mathrm{Tr}(A \otimes B) = \mathrm{Tr}(A)I;$

(f) $\dfrac{\partial}{\partial X}\mathrm{Tr}(X^2) = 2X^{\mathrm{T}};$

(g) $\dfrac{\partial}{\partial X}\mathrm{Tr}(X^{\mathrm{T}}AX) = AX + A^{\mathrm{T}}X;$

(h) $\dfrac{\partial}{\partial X}\mathrm{Tr}(X^{\mathrm{T}}X) = \dfrac{\partial}{\partial X}\|X\|_F^2 = 2X;$

(i) $\dfrac{\partial}{\partial X}\mathrm{Tr}(X^{\mathrm{T}}AXB) = AXB + A^{\mathrm{T}}XB^{\mathrm{T}}.$

GNN dynamics

The dynamic equation of the linear recurrent neural network for computing the inverse of a nonsingular matrix A is initiated by the error matrix $E(t) = AV(t) - I$.

Since $\varepsilon(V(t)) = \frac{1}{2}\|AV(t) - I\|_F^2 = \frac{1}{2}\mathrm{Tr}((AV(t) - I)^{\mathrm{T}}(AV - I))$, matrix calculus gives

$$\frac{\partial \varepsilon(V(t))}{\partial V} = \frac{1}{2}\frac{\partial \|AV(t) - I\|_F^2}{\partial V} = A^{\mathrm{T}}\left(AV(t) - I\right).$$

Now, using the general GNN design, corresponding GNN dynamics can be described as follows:

$$\begin{aligned} \frac{\mathrm{d}V(t)}{\mathrm{d}t} &= -\gamma A^{\mathrm{T}} AV(t) + \gamma A^{\mathrm{T}} \\ &= -\gamma A^{\mathrm{T}}\left(AV(t) - I\right), \ V(0) = V_0, \end{aligned} \tag{3}$$

where $V(t)$ is a matrix of activation state variables corresponding to the inverse matrix of A, and γ is a positive gain parameter.

Further, $V(0) = V_0$ denotes an arbitrary initial stage V_0.

GNN dynamics

The GNN design corresponding to the matrix equation

$$AXB = D$$

was investigated and applied in
[NEUCOM 2018a][P.S. Stanimirović, M.D. Petković, Gradient neural dynamics for solving matrix equations and their applications, Neurocomputing 306 (2018), 200–212.]

The error matrix is defined by $E(t) = D - AV(t)B$.

The scalar-valued norm-based error function is defined by the Frobenius norm of $E(t)$:

$$\varepsilon(t) = \varepsilon(V(t)) = \frac{1}{2}\|E(t)\|_F^2 = \frac{1}{2}\|D - AV(t)B\|_F^2.$$

According to $\frac{\partial}{\partial X}\mathrm{Tr}(AXB) = A^{\mathrm{T}}B^{\mathrm{T}}$ and the chain rule formula, the gradient of the objective function $\varepsilon(t)$ is equal to

$$\frac{\partial \varepsilon(V(t))}{\partial V} = -A^{\mathrm{T}}(D - AV(t)B)B^{\mathrm{T}} = -A^{\mathrm{T}}E(t)B^{\mathrm{T}}.$$

According to the general GNN dynamics, defined in Step2GNN, the GNN model for solving $AXB = D$ is defined as

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = \dot{V}(t) = \gamma A^{\mathrm{T}}(D - AV(t)B)B^{\mathrm{T}}. \tag{4}$$

The GNN model in (4) will be denoted by GNN(A, B, D).

GNN model

Convergence of the $GNN(A, B, D)$ model is defined in the next theorem.
The limiting value $\tilde{V}$ of $V(t)$ is determined by the choice of $V(0)$. For this purpose, it will be denoted by $\tilde{V}_{V(0)}$.

GNN model

Theorem 1

[NEUCOM 2018a] *Assume that the real matrices* $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$ *and* $D \in \mathbb{R}^{m\times q}$ *satisfy*

$$AA^{(1)}DB^{(1)}B = D. \tag{5}$$

Then the state matrix $V(t) \in \mathbb{R}^{n\times m}$ *of the GNN*(A, B, D) *model* (4) *satisfies:*

$$AV(t)B \to D, \quad t \to +\infty \tag{6}$$

for an arbitrary initial state matrix $V(0)$.

Proof. Firstly, Fulfillment of (12) for some $A^{(1)} \in A\{1\}$ and $B^{(1)} \in B\{1\}$ ensures solvability of the matrix equation $AXB = D$. The substitution $V(t) = \overline{V}(t) + A^{(1)}DB^{(1)}$ transforms the dynamics (4) into the equivalent form

$$\begin{aligned}\frac{\mathrm{d}\overline{V}(t)}{\mathrm{d}t} = \frac{\mathrm{d}V(t)}{\mathrm{d}t} &= \gamma A^{\mathrm{T}}\mathcal{F}\left(D - AV(t)B\right)B^{\mathrm{T}} \\ &= \gamma A^{\mathrm{T}}\mathcal{F}\left(D - A\left(\overline{V}(t) + A^{(1)}DB^{(1)}\right)B\right)B^{\mathrm{T}}.\end{aligned}$$

According to (12), it follows

$$\frac{\mathrm{d}\overline{V}(t)}{\mathrm{d}t} = \gamma A^{\mathrm{T}}\mathcal{F}\left(D - AA^{(1)}DB^{(1)}B - A\overline{V}(t)B\right)B^{\mathrm{T}} = -\gamma A^{\mathrm{T}}\mathcal{F}\left(A\overline{V}(t)B\right)B^{\mathrm{T}}. \tag{7}$$

The Lyapunov function candidate is

$$L(\overline{V}(t), t) := \frac{1}{2}\left\|\overline{V}(t)\right\|_F^2 = \frac{1}{2}\mathrm{Tr}\left(\overline{V}(t)^{\mathrm{T}}\,\overline{V}(t)\right). \tag{8}$$

Evidently, the inequality $L(\overline{V}(t), t) \geq 0$ holds for $\overline{V}(t) \neq 0$. According to (8), assuming (7) and using $\mathrm{dTr}\left(X^{\mathrm{T}}X\right) = 2\mathrm{Tr}\left(X^{\mathrm{T}}\mathrm{d}X\right)$ in conjunction with basic properties of the matrix trace function, one can express the time derivative of

$L(\overline{V}(t), t)$ as in the following:

$$\begin{aligned}\frac{\mathrm{d}L(\overline{V}(t),t)}{\mathrm{d}t} &= \mathrm{Tr}\left(\overline{V}(t)^{\mathrm{T}}\,\frac{\mathrm{d}\overline{V}(t)}{\mathrm{d}t}\right)\\ &= -\gamma\mathrm{Tr}\left[\overline{V}(t)^{\mathrm{T}}\,A^{\mathrm{T}}\mathcal{F}\left(A\overline{V}(t)B\right)B^{\mathrm{T}}\right]\\ &= -\gamma\mathrm{Tr}\left[B^{\mathrm{T}}\left(A\overline{V}(t)\right)^{\mathrm{T}}\mathcal{F}\left(A\overline{V}(t)B\right)\right]\\ &= -\gamma\mathrm{Tr}\left[\left(A\overline{V}(t)B\right)^{\mathrm{T}}\mathcal{F}\left(A\overline{V}(t)B\right)\right].\end{aligned} \tag{9}$$

Since the scalar-valued function $f(\cdot)$ is an odd and monotonically increasing, immediately follows $f(-x) = -f(x)$ and

$$f(x)\begin{cases} > 0 & \text{if } x > 0,\\ = 0 & \text{if } x = 0,\\ < 0 & \text{if } x < 0,\end{cases}$$

which in conjunction with $\gamma > 0$ implies for
$W(t) = A\overline{V}(t)B = A\left(V(t) - A^{(1)}DB^{(1)}\right)B = AV(t)B - D$

$$\begin{aligned}\frac{\mathrm{d}L(\overline{V}(t),t)}{\mathrm{d}t} &= -\gamma\mathrm{Tr}\left[\left(W^{\mathrm{T}}\mathcal{F}(W)\right)\right]\\ &= -\gamma\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}\,f(w_{ij})\begin{cases} < 0 & \text{if } A\overline{V}(t)B \neq 0,\\ = 0 & \text{if } A\overline{V}(t)B = 0.\end{cases}\end{aligned} \tag{10}$$

Now, the conclusion is

$$\frac{\mathrm{d}L(\overline{V}(t),t)}{\mathrm{d}t}\begin{cases} <0 & \text{if } W(t)\neq 0, \\ =0 & \text{if } W(t)=0. \end{cases} \tag{11}$$

This further implies:

- $\frac{\mathrm{d}L(\overline{V}(t),t)}{\mathrm{d}t} < 0$ at any non-equilibrium state $V(t)$ satisfying $W(t) = AV(t)B - D \neq 0$;
- $\frac{\mathrm{d}L(\overline{V}(t),t)}{\mathrm{d}t} = 0$ at the equilibrium state $V(t)$ satisfying $W(t) = AV(t)B - D = 0$.

According to the Lyapunov stability theory, $W(t) = AV(t)B - D$ globally converges to the zero matrix, from arbitrarily chosen $V(0)$. $\square$

GNN dynamics

Theorem 2

[NEUCOM 2018a] *Assume that the real matrices* $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$ *and* $D \in \mathbb{R}^{m\times q}$ *satisfy*

$$AA^{\dagger}DB^{\dagger}B = D. \tag{12}$$

Then the state matrix $V(t) \in \mathbb{R}^{n\times m}$ *of the GNN*(A, B, D) *model* (4) *satisfies:*

$$\begin{aligned} &AV(t)B \to D, \quad t \to +\infty \\ &\tilde{V}_{V(0)} = \lim_{t\to\infty} V(t) = A^{\dagger}DB^{\dagger} + V(0) - A^{\dagger}AV(0)BB^{\dagger} \end{aligned} \tag{13}$$

for an arbitrary initial state matrix $V(0)$.

Proof. In view of (4), the matrix $V_1(t) = A^\dagger AV(t)BB^\dagger$ satisfies

$$\frac{\mathrm{d}V_1(t)}{\mathrm{d}t} = \gamma A^\dagger A \frac{\mathrm{d}V(t)}{\mathrm{d}t} BB^\dagger = \gamma A^\dagger AA^{\mathrm{T}} \mathcal{F}\left(E(t)\right) B^{\mathrm{T}} BB^\dagger.$$

According to the basic properties of the Moore-Penrose inverse, it follows that $B^{\mathrm{T}} BB^\dagger = B^{\mathrm{T}}$, $A^\dagger AA^{\mathrm{T}} = A^{\mathrm{T}}$ which further implies

$$\begin{aligned}\frac{\mathrm{d}V_1(t)}{\mathrm{d}t} &= \gamma A^{\mathrm{T}} \mathcal{F}\left(E(t)\right) B^{\mathrm{T}} \\ &= \frac{\mathrm{d}V(t)}{\mathrm{d}t}.\end{aligned}$$

Consequently, $V_2(t) = V(t) - V_1(t)$ satisfies $\frac{\mathrm{d}V_2(t)}{\mathrm{d}t} = 0$, which implies

$$V_2(t) = V_2(0) = V(0) - V_1(0) = V(0) - A^\dagger AV(0)BB^\dagger, \quad t \geq 0. \tag{14}$$

Furthermore, according to Theorem 2, $AV(t)B \to D$ and $V_1(t)$ converges to

$$V_1(t) = A^\dagger (AV(t)B)B^\dagger \to A^\dagger DB^\dagger, \quad t \to +\infty.$$

Accordingly, having in mind (14), $V(t)$ is convergent and its equilibrium value is

$$\begin{aligned}V(t) = V_1(t) + V_2(t) \to \tilde{V} &= A^\dagger DB^\dagger + V_2(0) \\ &= A^\dagger DB^\dagger + V(0) - A^\dagger AV(0)BB^\dagger,\end{aligned}$$

which is just a confirmation of (13). □

GNN dynamics

It is known that

$$\|AXB - D\|_2 \geq \|AA^{\dagger}DB^{\dagger}B - D\|_2,$$

where the equality is valid if and only if

$$X = A^{\dagger}DB^{\dagger} + Y - A^{\dagger}AYBB^{\dagger}, \tag{15}$$

wherein Y is arbitrary matrix of appropriate dimensions.

Accordingly, (15) defines the set of least squares solutions to $AXB = D$.

In addition, $A^{\dagger}DB^{\dagger}$ is the unique minimizer of minimal norm between least squares minimizers:

$$\|A^{\dagger}DB^{\dagger}\|_2 \leq \|A^{\dagger}DB^{\dagger} + Y - A^{\dagger}AYB^{\dagger}B\|_2.$$

LS properties of the GNN(A,B,D) model

The following comments can be mentioned:
(i) Any solution

$$\tilde{V}_{V(0)} = A^{\dagger}DB^{\dagger} + V(0) - A^{\dagger}AV(0)BB^{\dagger}$$

generated by the $GNN(A, B, D)$ model (4) can be derived from the general LS solution

$$A^{\dagger}DB^{\dagger} + Y - A^{\dagger}AYBB^{\dagger}, \quad Y \text{ arbitrary}$$

to the matrix equation $AXB = D$ by replacing the arbitrary matrix Y from the LS solution by the initial state matrix $V(0)$.

(ii) The Moore-Penrose solution (i.e., the minimal norm least squares solution) $\tilde{V}_0 = A^{\dagger}DB^{\dagger}$ to the matrix equation $AXB = D$ can be generated by $GNN(A, B, D)$ using the zero initial state $V(0) = 0$.

Corollary 1

Assume that the real matrices $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$ and $D \in \mathbb{R}^{m\times q}$ satisfy (**??**). *Further, let an odd and monotonically increasing function $f(\cdot)$ be used to define the array activation function $\mathcal{F}(\cdot)$ and the design parameter γ satisfy $\gamma > 0$. Let $\tilde{V}_{V(0)}$ denotes the limit value $\tilde{V} = \lim_{t\to\infty} V(t)$ corresponding to the initial state $V(0)$. Then the equilibrium state matrix $\tilde{V}_{V(0)}$ of $GNN(A, B, D)$ satisfies*

$$\left\{A\,\tilde{V}_{V(0)}\,B|\ V(0) \in \mathbb{R}^{n\times m}\right\} = \{D\}. \tag{16}$$

for each $V(0) \in \mathbb{R}^{n\times m}$.

Lecture 6:, Tutorial 3:

Lecture 6:
- Design parameters in GNN evolutionary design.
- Properties of activation functions in RNN, overview of commonly used activation functions (AFs): linear, bipolar sigmoid, power AF, power-sigmoid, hyperbolic sine, sign-bi-power, tunable AF.
- Influence of gain parameters and activations functions on the convergence speed.

Tutorial 3:
- Implementation of ZNN models for solving various time-varying matrix equations.
- Applications in computing generalized inverses and solving linear systems.

Main activation functions

The nonlinear gradient-based neural dynamics exploits acceleration by the nonlinear activation function array $\mathcal{F}()$:

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = \dot{V}(t) = \gamma \mathcal{F}\left(\frac{\partial \varepsilon(t)}{\partial V}\right).$$

An activation function $\mathcal{F}$ is defined on a real matrix $C = (c_{ij}) \in \mathbb{R}^{m \times n}$ by the element-wise application $\mathcal{F}(C) = (f(c_{ij}))$ of an odd and monotonically increasing function f.Monotonically increasing and odd activation functions:
(1) Linear function:

$$f(x) = x.$$

(2) Bipolar-sigmoid activation function:

$$f(x) = \frac{1 + exp(-\varrho)}{1 - exp(-\varrho)} \cdot \frac{1 - exp(-\varrho x)}{1 + exp(-\varrho x)}, \ q > 2.$$

(3) Power-sigmoid function:

$$f(x) = \begin{cases} x^{\rho}, & if \ |x| \geq 1 \\ \dfrac{1 + exp(-\varrho)}{1 - exp(-\varrho)} \cdot \dfrac{1 - exp(-\varrho x)}{1 + exp(-\varrho x)}, & otherwise \end{cases},$$

where $\varrho > 2, \ \rho \geq 3$.

Main activation functions

(4) Sign-bi-power function (Li function):

$$f(x) = \frac{1}{2}|x|^r \text{sign}(x) + \frac{1}{2}|x|^{\frac{1}{r}} \text{sign}(x),$$

where $0 < r < 1$.
(5) Tunable sign-bi-power function (Tunable function):

$$f(x) = \frac{1}{2}k_1|x|^r \text{sign}(x) + \frac{1}{2}k_2 x + \frac{1}{2}k_3|x|^{\frac{1}{r}} \text{sign}(x),$$

where $k_1 > 0,\ k_2 > 0,\ k_3 > 0,\ 0 < r < 1$.

Main activation functions

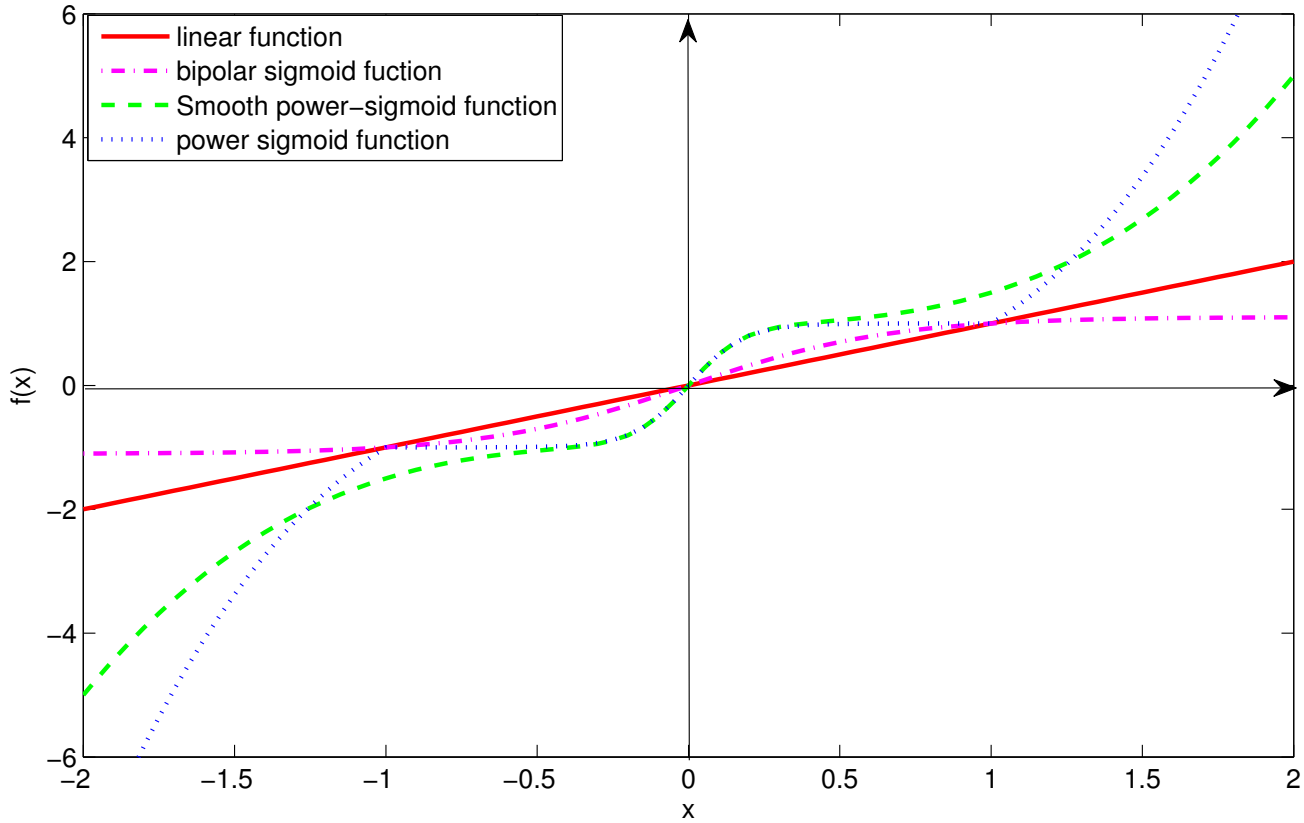


Figure 1: Behavior of the four basic types of activation functions

Accordingly, the nonlinear GNN model for solving $AXB = D$ is defined as

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = \dot{V}(t) = \gamma A^{\mathrm{T}} \mathcal{F}(D - AV(t)B)B^{\mathrm{T}}. \tag{17}$$

Main activation functions

The **sign** activation funtion

$$\text{sign}(u) = \begin{cases} 1, & u > 0 \\ 0, & u = 0 \\ -1, & u < 0 \end{cases} \tag{18}$$

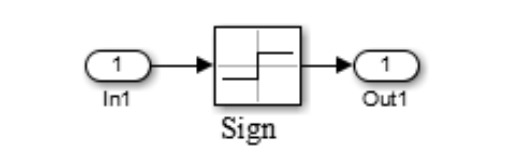


Figure 2: Simulink block of sign activation function (18).

Main activation functions

Previous nonlinear AFs in ZNN formula are used to accelerate the convergence speed in finite or predefined time. Some of them are referred as follows.
Li or sign-bi-power (**sbp**) activation function

$$\mathrm{sbp}(u) = \mathrm{sgn}^{\mathrm{q}}(\mathrm{u}) + \mathrm{sgn}^{\frac{1}{\mathrm{q}}}(\mathrm{u}), \tag{19}$$

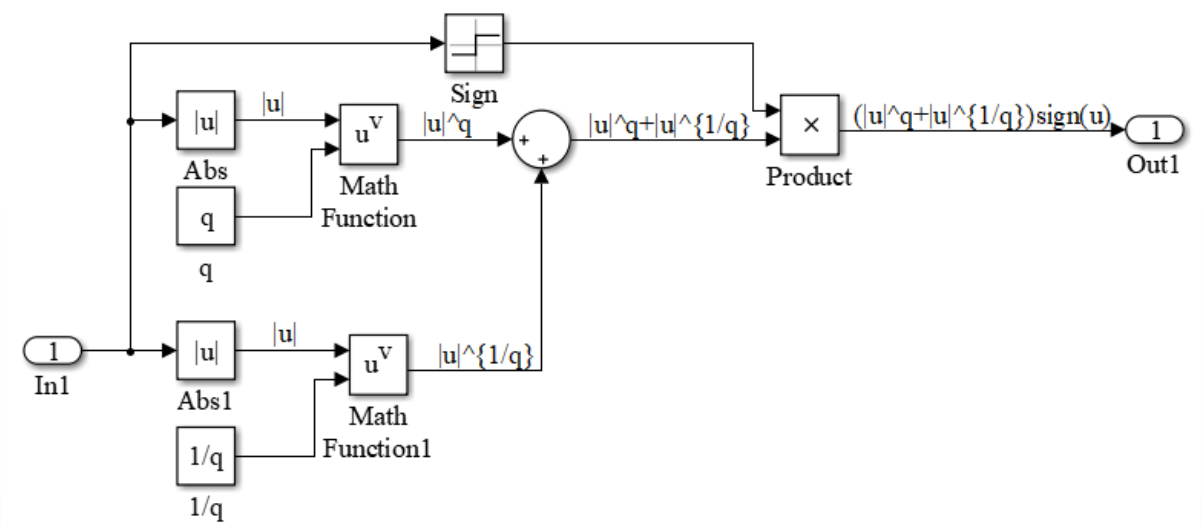


Figure 3: Simulink block of Li activation function (19).

where

$$\mathrm{sgn}^{\mathrm{q}}(u) = |\mathrm{u}|^{\mathrm{q}}\mathrm{sign}(u) = \begin{cases} |\mathrm{u}|^{\mathrm{q}}, & u > 0 \\ 0, & u = 0 \\ -|\mathrm{u}|^{\mathrm{q}}, & u < 0 \end{cases}, \quad q \in (0,1). \tag{20}$$

Main activation functions

- The **tunable** activation function

$$G(u) = (|u|^q + |u| + |u|^{\frac{1}{q}})sign(u), \tag{21}$$

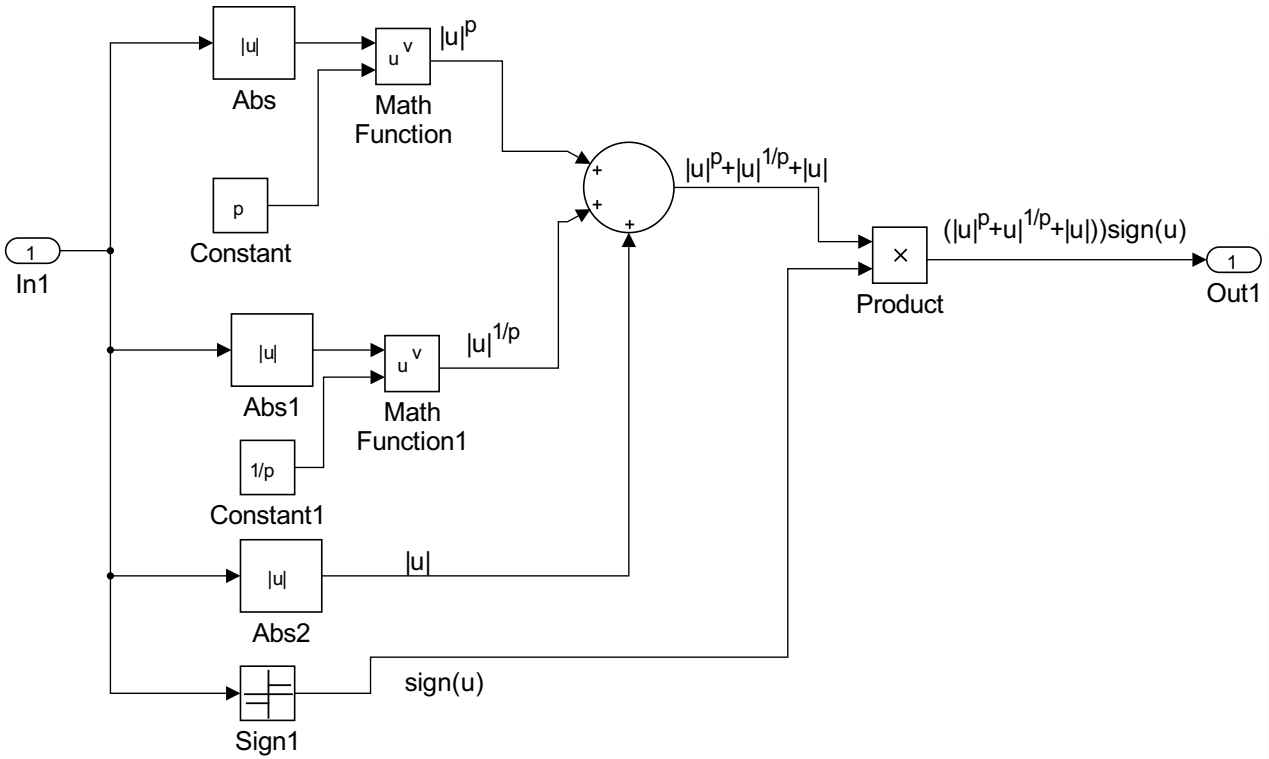


Figure 4: Simulink block of Tunable activation function (21).

Main activation functions

- The versatile activation function (**VAF**)

$$\mathrm{G(u)} = (a_1|\mathrm{u}|^{\mathrm{q}} + a_2|\mathrm{u}|^{\mathrm{w}})\mathrm{sign}(u) + a_3\mathrm{u} + a_4\mathrm{sign(u)}, \tag{22}$$

where $a_1, a_2 > 0$, $a_3, a_4 \geq 0$ and $\mathrm{w} > 1$.

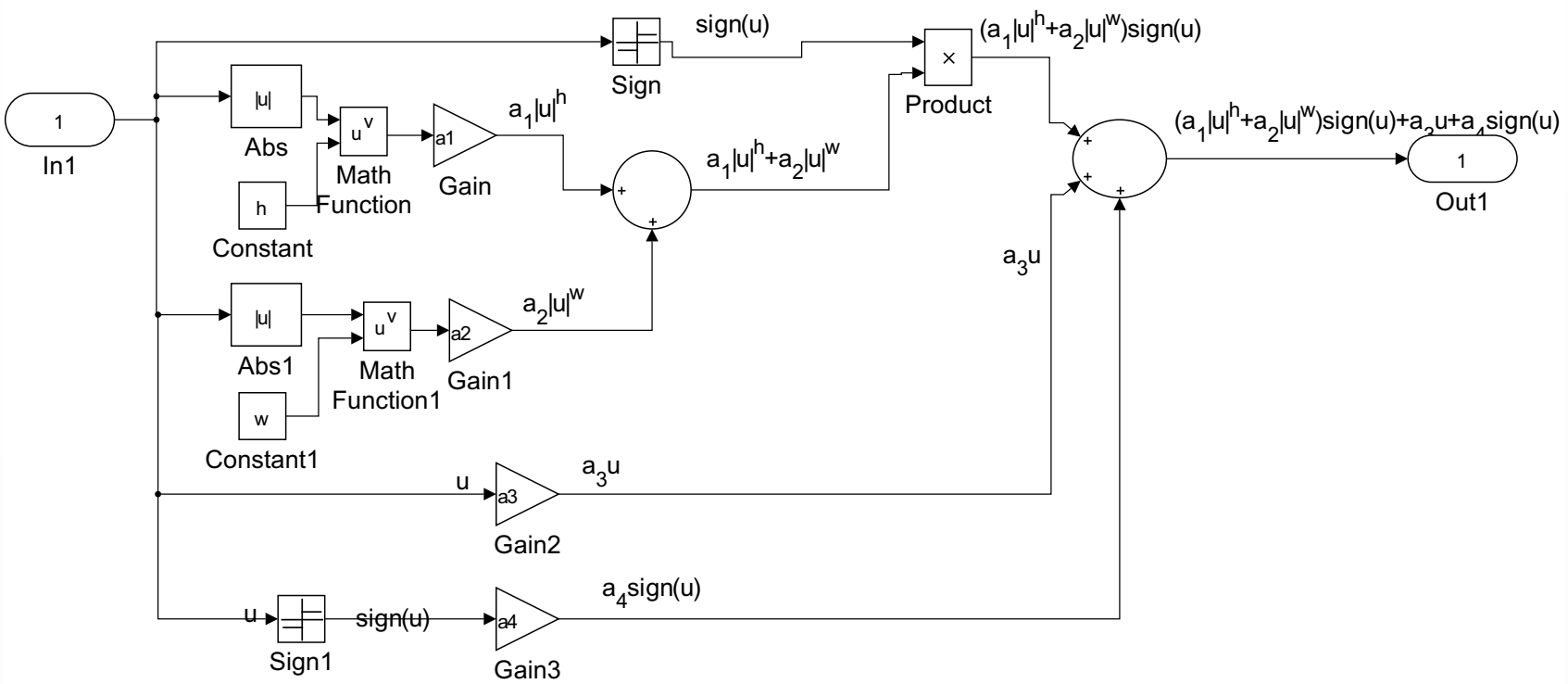


Figure 5: Simulink block of versatile activation function (VAF) (22).

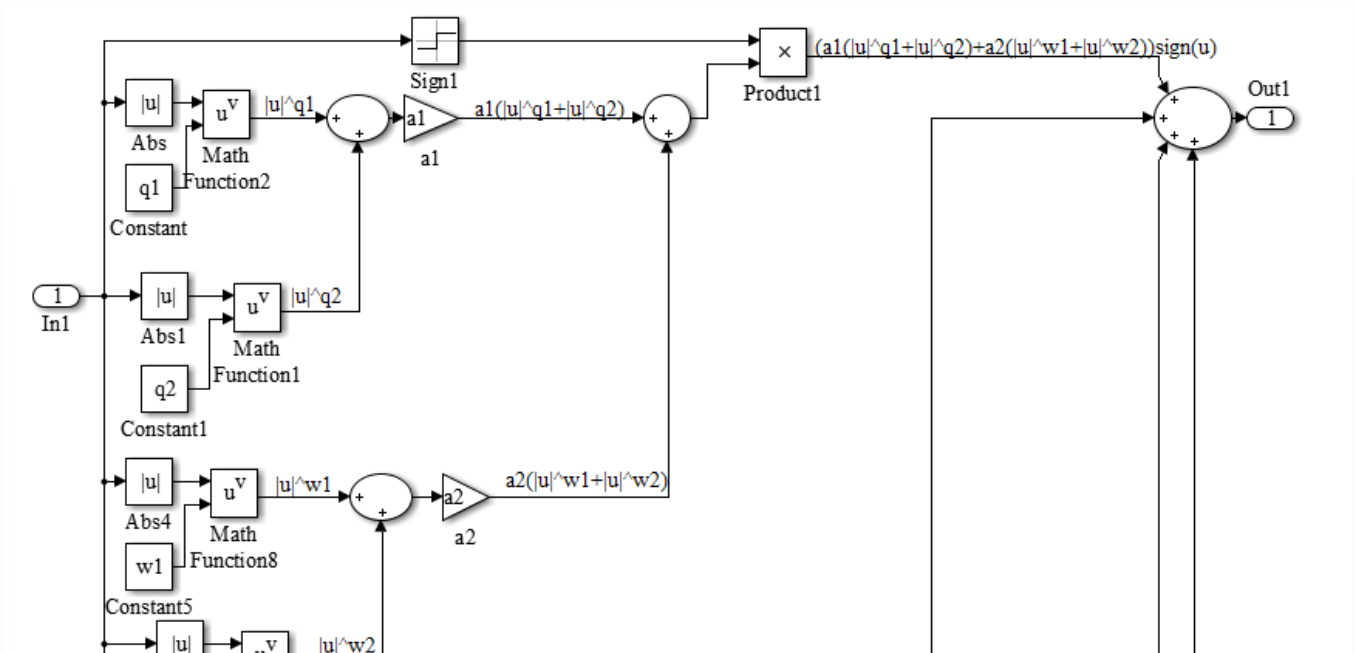
Main activation functions

- The extended versatile activation function (**EVAF**) of order m

$$G_m(u) = \left(a_1 \sum_{k=1}^{m} |u|^{q_k} + a_2 \sum_{k=1}^{m} |u|^{w_k} \right) \operatorname{sign}(u) + a_3 u + a_4 \sinh(u) + a_5 (\exp(|u|) - 1) \operatorname{sign}(u), \tag{23}$$

where $q_k \in (0,1)$, $w_k > 1$, a_1, a_2, a_3, a_4 satisfy the same constraints as in previous VAF (22), $a_5 \geq 0$ and $k = 1, 2, \ldots, m$. Suitable values for the remaining parameters are $0 < q_1 \leq q_2 \leq \cdots \leq q_m < 1$ and $1 < w_1 \leq w_2 \leq \cdots \leq w_m$. For the purpose of this tutorial we will use and design the EVAF activation function order 2:

$$G_2(u) = (a_1 (|u|^{q_1} + |u|^{q_2}) + a_2 (|u|^{w_1} + |u|^{w_2})) \operatorname{sign}(u) + a_3 u + a_4 \sinh(u) + a_5 (\exp(|u|) - 1) \operatorname{sign}(\tag{24}$$



Main activation functions

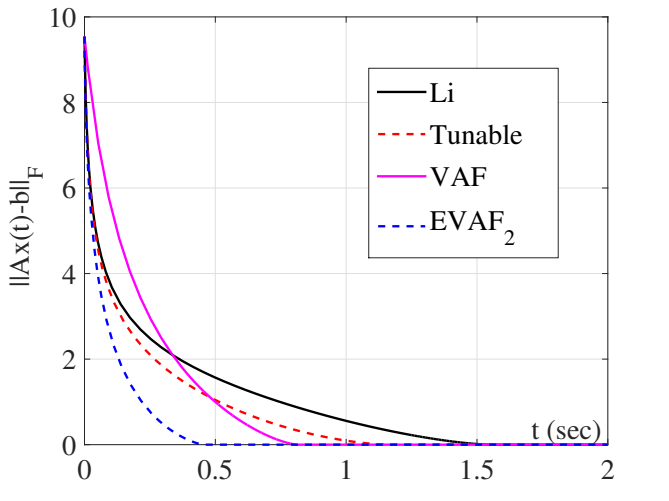


Figure 7: Error comparison.

It is observable from Figure 7 that the extended versatile AF (24) includes faster convergence property against previous activation functions because of Frobenius norm $||E(t)||_F = ||\mathrm{Ax(t)} - \mathrm{b}||_F$ vanish to zero in shorter time.

Tutorial 1:

Tutorial 1:
- Simulink as an efficient tool for agile software development.
- Simulink implementation of GNN for solving the general linear matrix equations AXB=C.
- Simulink for GNN design for computing the matrix inverse, left and right inverse and the Moore-Penrose generalized inverse.
- Simulink implementation of GNN models for solving systems of linear equations.

About Simulink modeling

What is a Simulink model?
A Simulink model is a graphical representation of a system created within the Simulink environment, which is an add-on to MATLAB.
It allows users to design, simulate, and test systems using a block diagram approach, where different components are represented as blocks connected by signals.
Users can create models by dragging and dropping blocks from libraries, representing functions, components, and systems.

Why Simulink is used?
Simulink is used to model, simulate, and analyze dynamic systems.

Design. Simulate. Deploy.
Simulink is a block diagram environment used to design systems, simulate before moving to hardware, and develop without writing code.

Why use Simulink instead of MATLAB?
You can also create custom blocks using MATLAB functions or other Simulink models.
Simulink blocks provide a visual representation of your system, which can help you to verify its logic and behavior.
On the other hand, MATLAB code requires you to write and edit text commands, which can be more complex and error-prone.

Creating a MATLAB Simulink model involves several steps, including defining the system you want to simulate, building the model using blocks, and configuring the parameters.

Simulink implementation of GNN dynamics

- The **Constant** block creates a complex or real constant value signal. The use of this block is to give a constant signal input and can generate scalar, vector, or matrix output.
- The **Product** block generates the result of multiplying two inputs and can be of two scalars, product of a scalar and a nonscalar, or product of two nonscalars of appropriate dimensions.
- The **Product1** block can be derived from **Product** block and it is used for the multiplication of two (or more) matrices.
- The **Abs** block returns the absolute value of the input.
- The **Sum** block returns the sum of two inputs and can be used for more than two inputs.
- The **Gain** block multiplies the input by a constant value.
- The **Display** block represents the value of the input.
- The **Scope** block is used for the presentation of time domain signals.
- The **To Workspace** block is used to save the input to specified time series, array, or structure in a workspace in Matlab.

GNN dynamics

- The **Integrator** block outputs the value of the integral of its input signal with respect to time.
- The **Math Function** block includes different mathematical functions such as logarithmic, exponential, power, modulus functions and etc.
- The **Subsystem** block includes a subset of blocks within a model or system and can be used for the presentation of a virtual subsystem or a nonvirtual subsystem.
- The **Interpreted Matlab Functions** block applies specified Matlab function or expression to the input data.

GNN dynamics

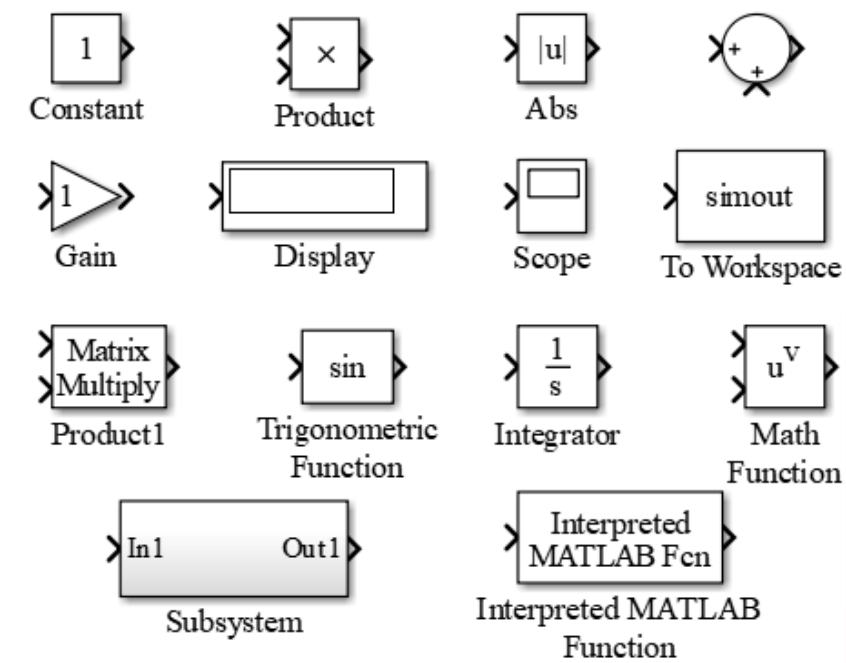


Figure 8: Basic Simulink blocks.

GNN dynamics

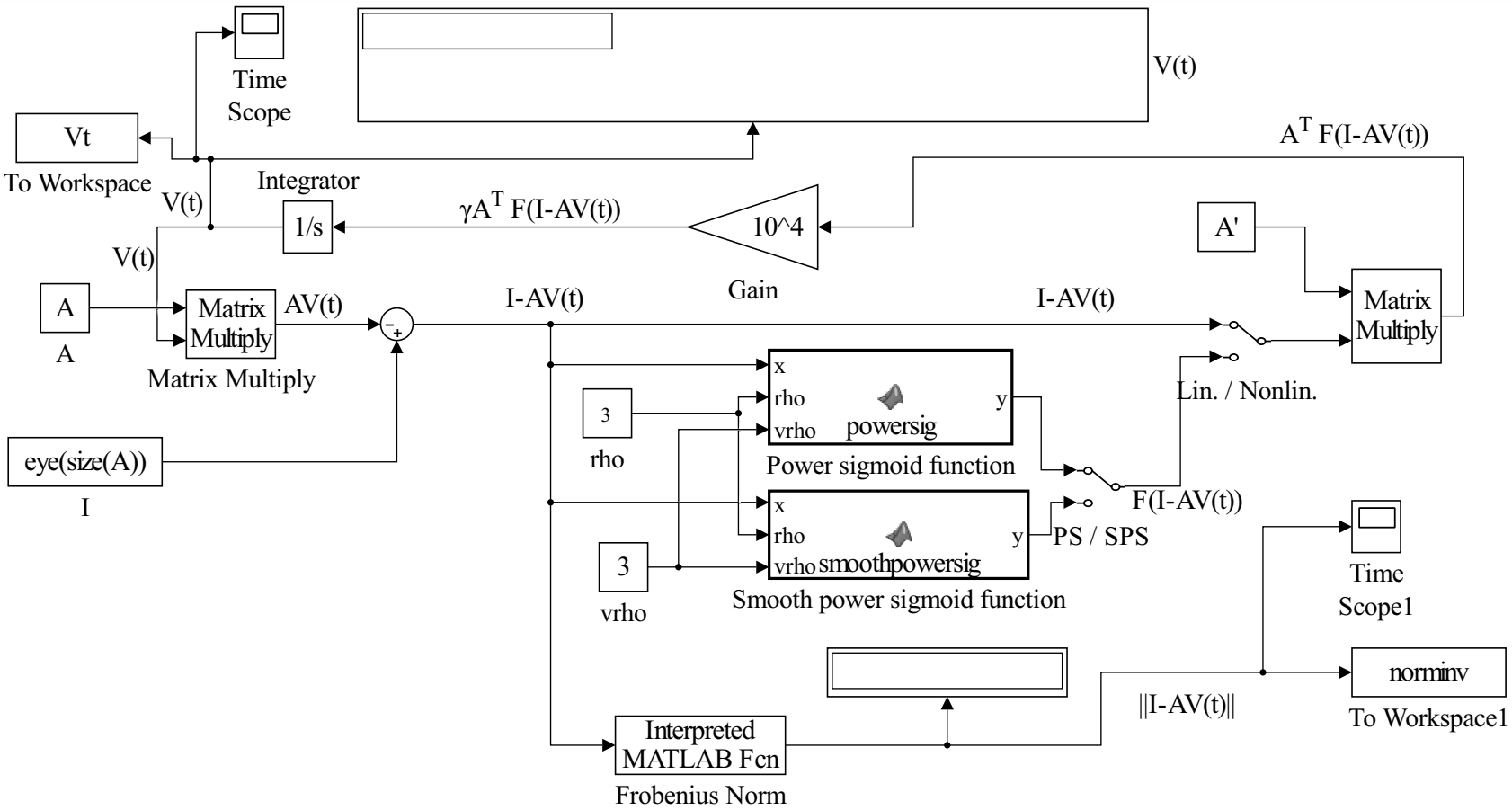


Figure 9: Simulink implementation of (3) for computing A^{-1}.

GNN dynamics

Consider the matrix

$$A = \begin{bmatrix} 1 & 10 & 10 \\ 3 & 10 & 10 \\ 6 & 2 & 5 \end{bmatrix}$$

generated by the command `A = randi([1 10],3,3)`. Using $\gamma = 10^4$ and the zero initial state Simulink generates elemenwise trajectories as in Figure 10 in the time interval [0.0.1] (left) and in [0.0.01] (right).

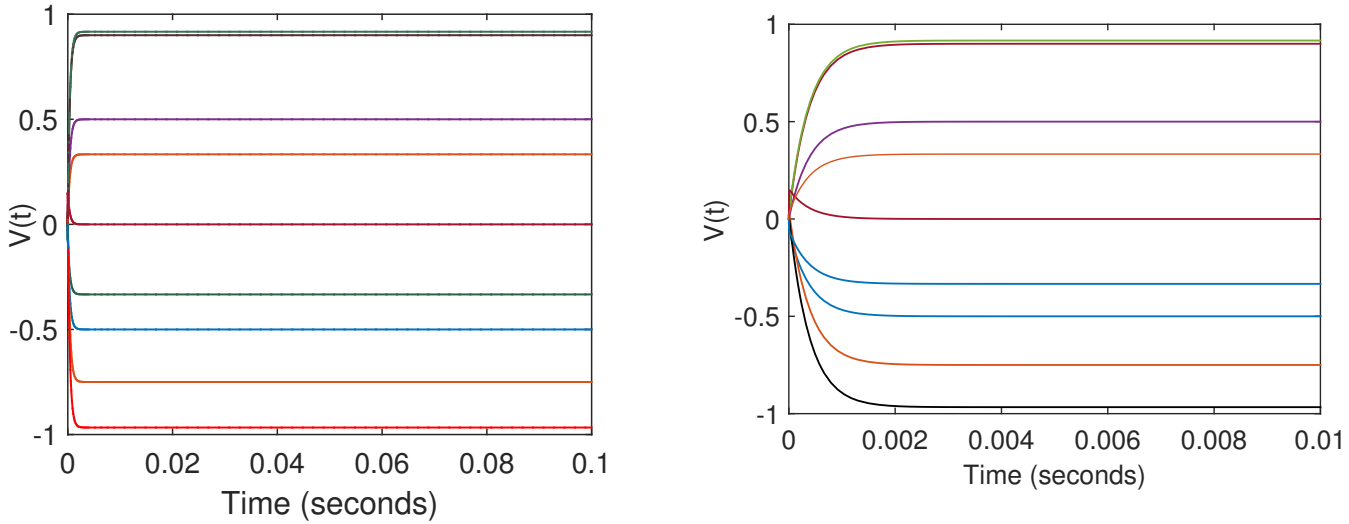


Figure 10: Elementwise trajectories of $V(t)$ in $[0, 0.1]$ (left) and in [0.0.01] (right).

GNN dynamics

The error norm $\|I - AV(t)\|_F$ in the time interval $[0.10^{-5}]$ is presented in Figure 11, left, for $\gamma = 10^3$,$\gamma = 10^4$ and $\gamma = 10^6$. The error norm in the time interval $[0.10^{-1}]$ is presented in Figure 11, right, for $\gamma = 10^2$,$\gamma = 10^3$ and $\gamma = 10^4$.

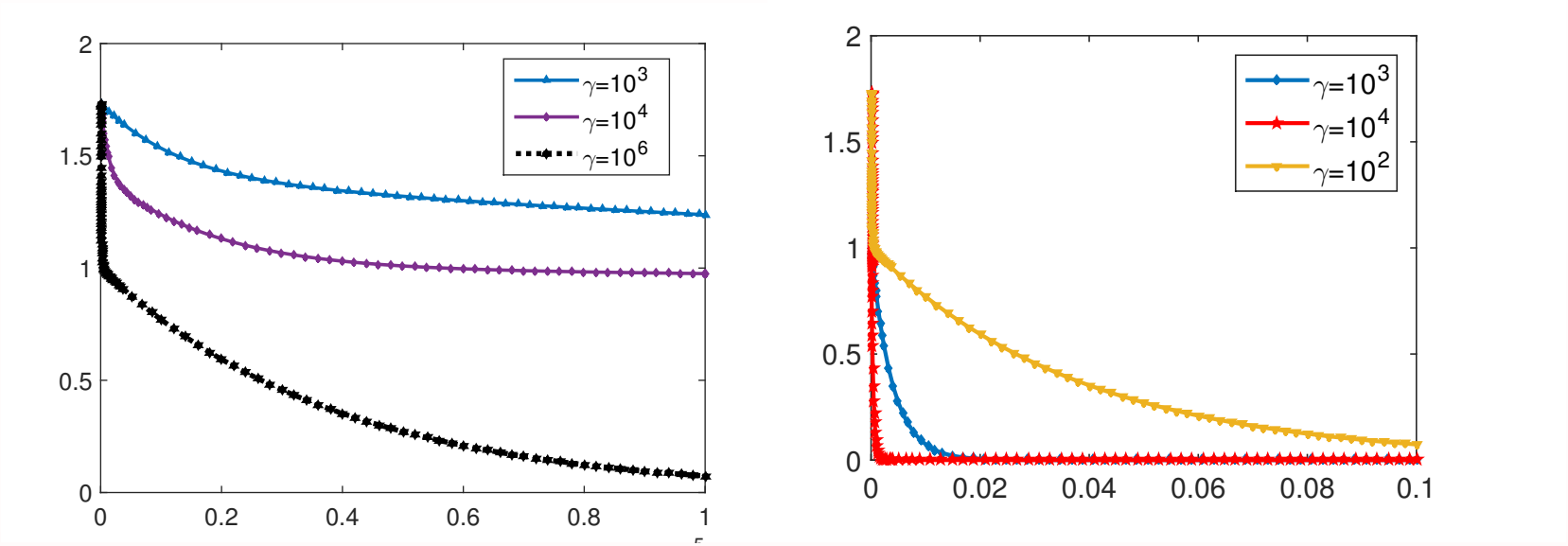
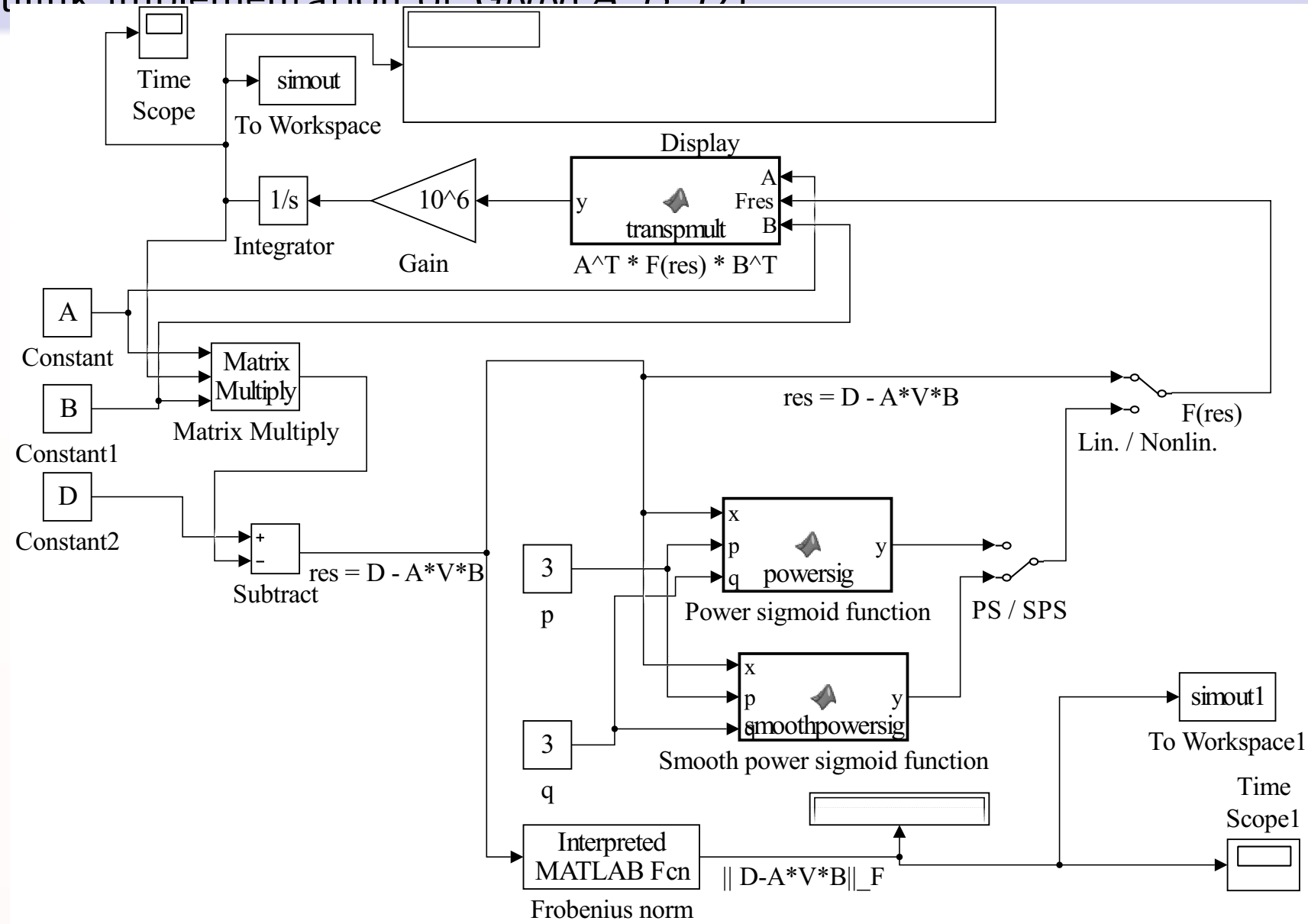


Figure 11: $\|I - AV(t)\|_F$ for different γ: in $[0, 10^{-5}]$ (left) and $[0, 10^{-1}]$ (right).

Graphs in Figure 11 confirm that larger values of the gain parameter γ initiate faster convergence $\|I - AV(t)\| \to 0$.

Simulink implementation of *GNN(A, B, D)*



Numerical examples for GNN(A,B,D) model

Consider the following matrices

$$A = \begin{bmatrix} -8 & 8 & -4 \\ 11 & 4 & -7 \\ 1 & -4 & 3 \\ 0 & 12 & -10 \\ 6 & 12 & -12 \end{bmatrix}, \quad B = \begin{bmatrix} 12 & -4 & -16 \\ -35 & -10 & 45 \\ 2 & -5 & -3 \\ 14 & 17 & -17 \end{bmatrix},$$

$$D = \begin{bmatrix} -84 & 2524 & 304 \\ -2252 & -623 & 2897 \\ 484 & -885 & -701 \\ -1894 & 2278 & 2652 \\ -2778 & 1524 & 3750 \end{bmatrix}.$$

Numerical examples for GNN(A,B,D) model

All three matrices are of rank $r = 2$ and satisfy the condition $AA^{\dagger}DB^{\dagger}B = D$.
We use the GNN(A, B, D) model (4) to compute the minimum norm least squares solution

$$X = A^{\dagger}DB^{\dagger} = \begin{bmatrix} \frac{197968}{75725} & \frac{2043}{1165} & \frac{35007}{15145} & -\frac{479239}{75725} \\ -\frac{17404}{3029} & \frac{660}{233} & -\frac{11337}{3029} & \frac{21877}{3029} \\ \frac{296594}{75725} & -\frac{3431}{1165} & \frac{71137}{30290} & -\frac{592049}{151450} \end{bmatrix}$$

$$\approx \begin{bmatrix} 2.6143 & 1.75365 & 2.31146 & -6.32868 \\ -5.74579 & 2.83262 & -3.74282 & 7.22252 \\ 3.91672 & -2.94506 & 2.34853 & -3.9092 \end{bmatrix}.$$

of the matrix equation $AXB = D$.

The parameters of the $GNN(A, B, D)$ model are $\gamma = 10$ and $V(0) = 0$.

Numerical examples for GNN(A,B,D) model

Elementwise trajectories of $V(t)$ are shown on the left graphs in Figure 1. It is evident that these trajectories follow a usual convergence behaviour towards the Equilibrium states, which have no further tendency to change in time.
Right graphs in Figure 1 show the error norm $\|E(t)\|_F = \|D - AV(t)B\|_F$ when both linear and non-linear activation functions are used.

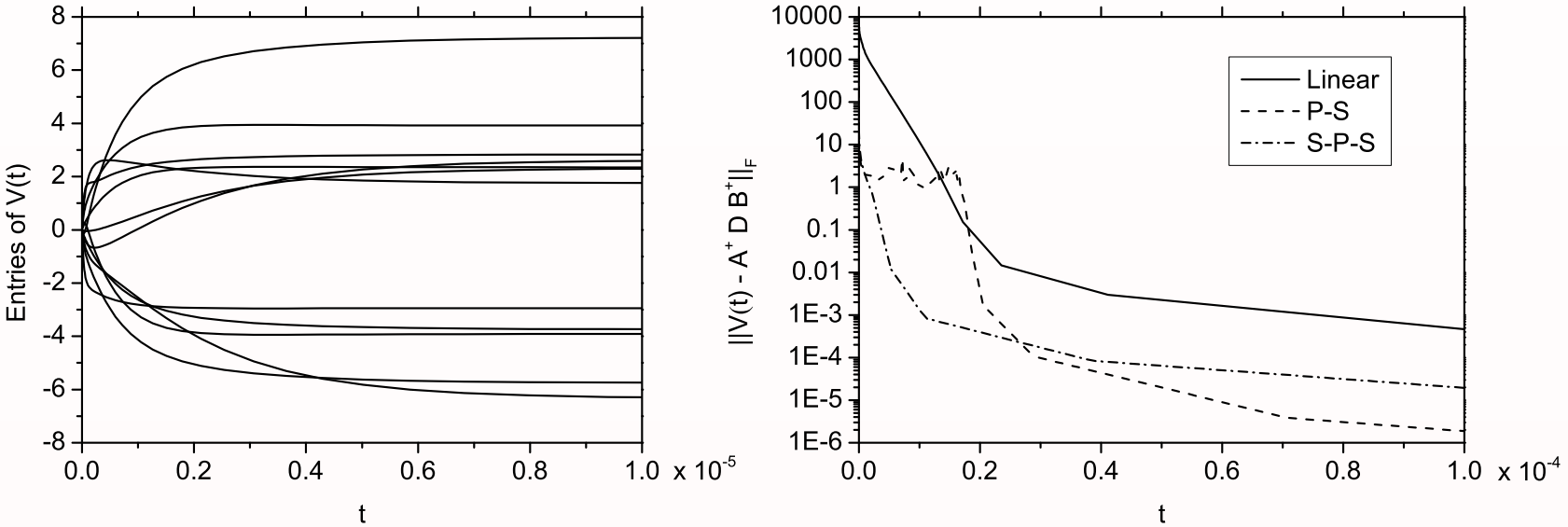


Figure 12: Elementwise trajectories of $GNN(A, B, D)$ (left) and the error norm trajectories for different activations (right).

Particular GNN(A,B,D) models

(a) $GNN(A, I, I)$ model is aimed to solving $AX = I$ and uses the error matrix

$$E(t) = AV(t) - I.$$

It was proposed in 1993.
Its dynamics can be expressed as

$$\dot{V}(t) = \frac{dV(t)}{dt} = -\gamma A^{\mathrm{T}} \mathcal{F}\left(AV(t) - I\right), \; V(0) = V_0. \tag{25}$$

Corollary 2

The general solution of the $\mathrm{GNN}(A, I, I)$ *model, given in* (25)*, is equal to*

$$\tilde{V}_{V(0)} = A^{\dagger} + V(0) - A^{\dagger} AV(0).$$

The $\mathrm{GNN}(A, I, I)$ model can be used in:
- finding the inverse of nonsingular A, starting from arbitrary $V(0)$;
- approximating the left inverse of a full-column rectangular matrix A, starting from arbitrary $V(0)$;
- computing the pseudoinverse of rank-deficient matrices under the zero initial condition $V(0) = 0$.

Analyse differerent $V(0)$.

Particular GNN(A,B,D) models

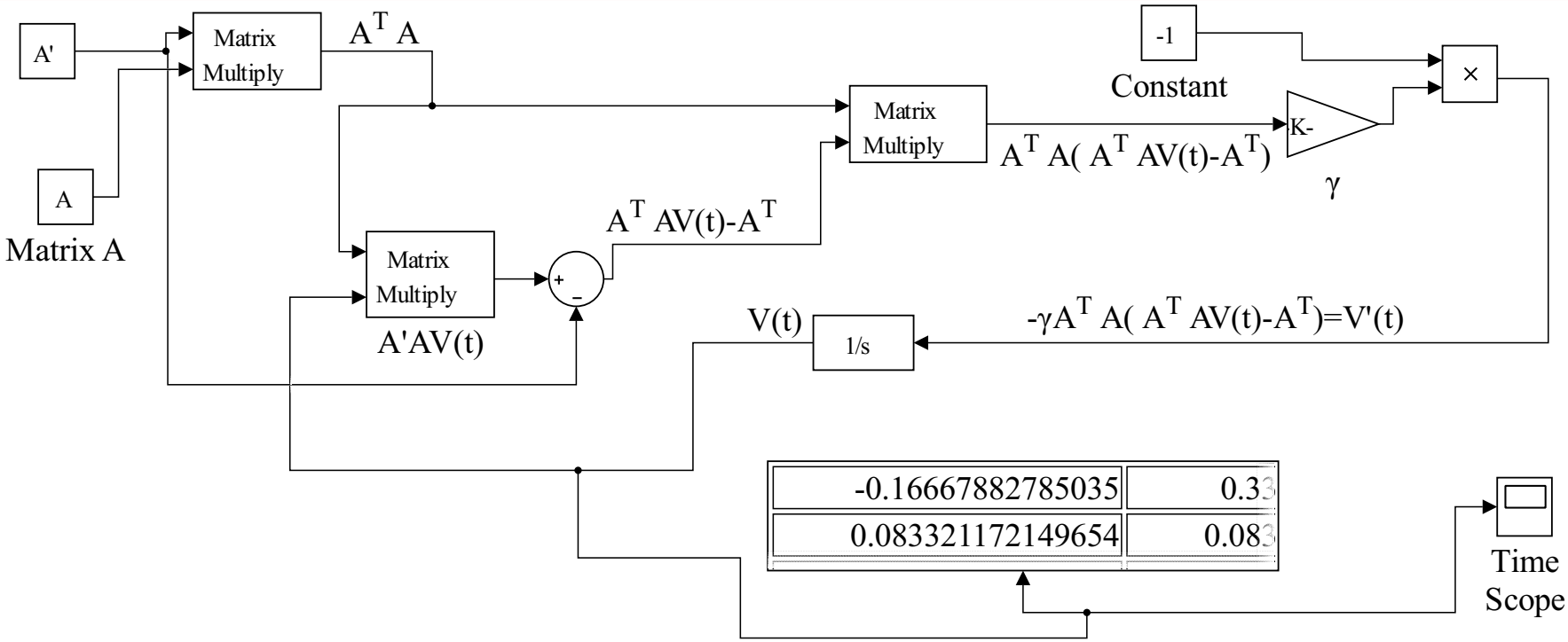
(b) The Moore-Penrose inverse $A^{\dagger}$ satisfies $A^{\mathrm{T}}AA^{\dagger} - A^{\mathrm{T}} = O$.
So, the scalar goal function is equal to

$$\varepsilon(t) = \frac{1}{2}\|A^{\mathrm{T}}AV(t) - A^{\mathrm{T}}\|_F^2.$$

Therefore, the linear GNN model (called LGNN-L) for approximating the Moore-Penrose inverse was defined as

$$\dot{V}(t) = -\gamma\frac{\partial\varepsilon(V(t))}{\partial V} = -\gamma A^{\mathrm{T}}A\left(A^{\mathrm{T}}AV(t) - A^{\mathrm{T}}\right). \tag{26}$$

Simulink implementation of GNN (26) is presented in Figure 13.



Particular GNN(A,B,D) models

(c) GNN model for solving the matrix equation $AXA = A$ was introduced as the GNN-MP model in
[NEPL2018][P.S. Stanimirović, M.D. Petković, D. Gerontitis, *Gradient neural network with nonlinear activation for computing inner inverses and the Drazin inverse*, Neural Processing Letters 48 (2018), 109–133.

The $GNN(A, A, A)$ model is defined from the error $E(t) = A - AV(t)A$ as

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = \gamma A^{\mathrm{T}} \mathcal{F}\left(A - AV(t)A\right) A^{\mathrm{T}} = \gamma A^{\mathrm{T}} \mathcal{F}\left(E(t)\right) A^{\mathrm{T}}, \quad V(0) = V_0.$$

Corollary 3

[NEPL2018] *Let $A \in \mathbb{R}^{m\times n}$ be arbitrary real matrix. Then:*
1. The matrix of activation state variables $V(t) \in \mathbb{R}^{n\times m}$ in the $GNN(A, A, A)$ is convergent when $t \to +\infty$ and its limiting value satisfies

$$\tilde{V}_{V(0)} = A^{\dagger} + V(0) - A^{\dagger} AV(0)AA^{\dagger}.$$

2. $\left\{ \tilde{V}_{V(0)} |\ V(0) \in \mathbb{R}^{n\times m} \right\} = A\{1\}.$

The proof of part 2. follows from

$$A\{1\} = \left\{ A^{(1)} + Y - A^{(1)} AYAA^{(1)} \right\},$$

for arbitrary $A^{(1)}$.

Particular GNN(A,B,D) models

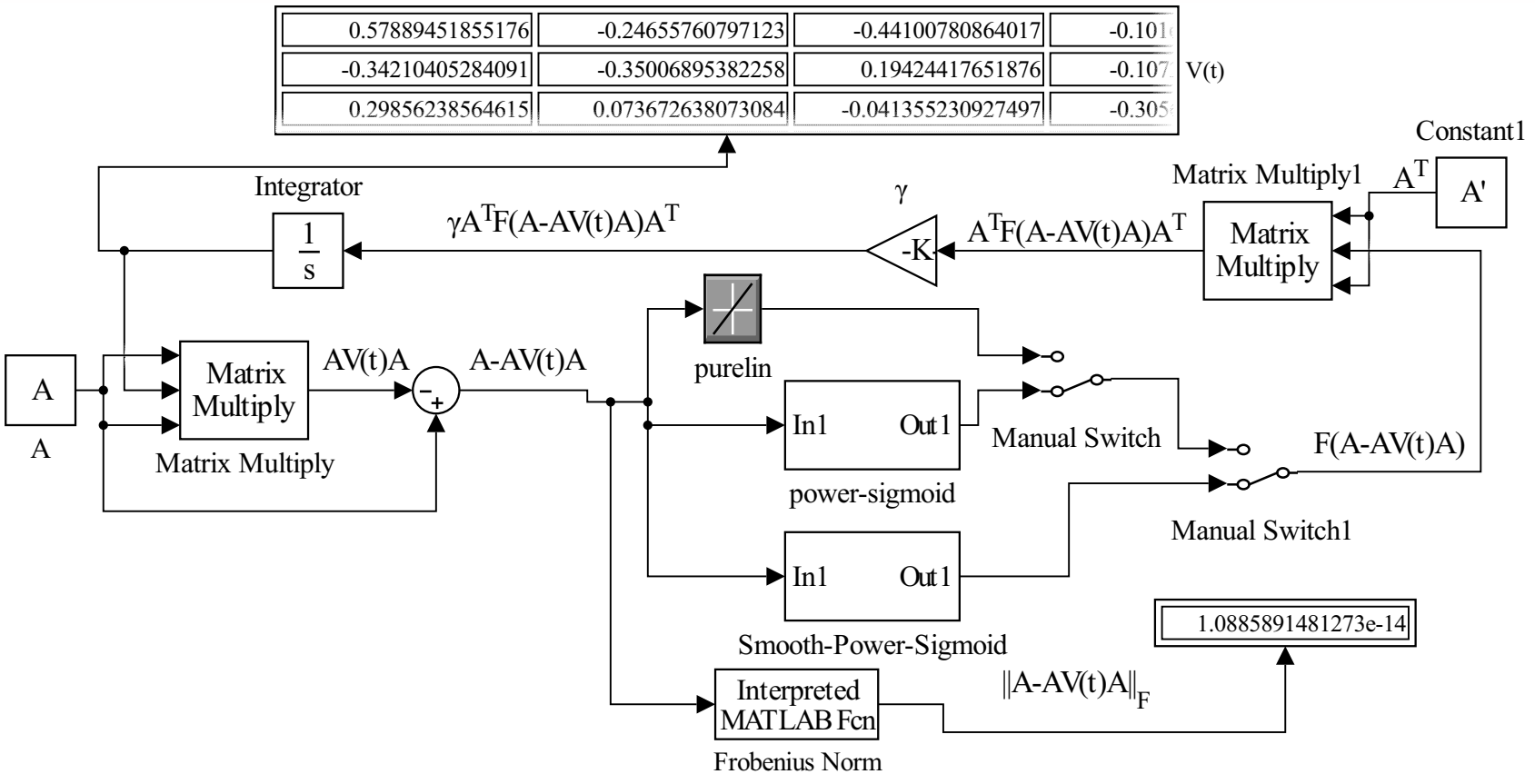


Figure 14: Simulink implementation of the GNN(A, A, A) model.

Tutorial 5:

Tutorial 5:
Implementation of GNN and ZNN models for solving various time-varying matrix equations.
Applications in computing generalized inverses and solving linear systems.

Particular GNN(A,B,D) models

Proposition 1

Let $A \in \mathbb{C}_r^{m\times n}$ be an arbitrary matrix, $0 < s \leq r$ and $A = PQ$ is a full-rank factorization of A. The following general representations for some classes of generalized inverses are valid:

$$A\{2\}_s = \{F(GAF)^{-1}G \mid F \in \mathbb{C}^{n\times s},\ G \in \mathbb{C}^{s\times m},\ \operatorname{rank}(GAF) = s\};$$
$$A\{2\} = \cup_{s=0}^{r} A\{2\}_s;$$
$$A\{2,4\}_s = \left\{(VA)^{\dagger}V \mid V \in \mathbb{C}^{s\times m},\ VA \in \mathbb{C}_s^{s\times n}\right\};$$
$$A\{2,3\}_s = \left\{U(AU)^{\dagger} \mid U \in \mathbb{C}^{n\times s},\ AU \in \mathbb{C}_s^{m\times s}\right\};$$
$$A\{1,2\} = A\{2\}_r;\ A\{1,2,4\} = A\{2,4\}_r;\ A\{1,2,3\} = A\{2,3\}_r$$
$$A^{\dagger} = Q^*(P^*AQ^*)^{-1}P^*;$$

A square matrix A has a group inverse if and only if QP is nonsingular, in which case

$$A^{\#} = P(QP)^{-2}Q.$$

Particular GNN(A,B,D) models

(d) GNN for solving GAX=G.
Let $A \in \mathbb{C}_r^{m\times n}$ be given and $G \in \mathbb{C}_s^{n\times m}$, $0 < s \leq r$, be appropriately chosen matrix such that

$$GA(GA)^{\dagger}G = G.$$

This condition is satisfied if A and G fulfil

$$\mathrm{rank}(GA) = \mathrm{rank}(AG) = \mathrm{rank}(G).$$

The GNN model for solving the matrix equations $GAX = G$ is denoted by $GNN(GA, I, G)$.
The underlying error matrix is $E(t) = GAV(t) - G$ and its neural dynamics is

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = -\gamma(GA)^{\mathrm{T}}\mathcal{F}\left(GAV(t) - G\right), \;\; V(0) = V_0.$$

Particular GNN(A,B,D) models

Corollary 4

Assume that the real matrices $A \in \mathbb{R}_r^{m\times n}$, $G \in \mathbb{R}_s^{n\times m}$ *satisfy* $0 < s \leq r$ *and*

$$\operatorname{rank}(GA) = \operatorname{rank}(G) \Longleftrightarrow GA(GA)^{\dagger} G = G.$$

Then:

(i) *The unknown matrix* $V(t)$ *of the model* $GNN(GA, I, G)$

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = -\gamma\,(GA)^{\mathrm{T}}\,(GAV(t) - G)\,, \quad V(0)\ \textit{arbitrary} \tag{27}$$

is convergent when $t \to +\infty$ *and has the limit value*

$$\tilde{V}_{V(0)} = (GA)^{\dagger} G + V(0) - (GA)^{\dagger} GAV(0). \tag{28}$$

(ii) *In particular,* $V(0) = 0$ *initiates* $\tilde{V}_0 = (GA)^{\dagger} G = A^{(2,4)}_{\mathcal{R}((GA)^{\mathrm{T}}),\mathcal{N}(G)}$.

Particular GNN(A,B,D) models

The Simulink implementation of GNN(GA, I, G) is illustrated in Figure 15.

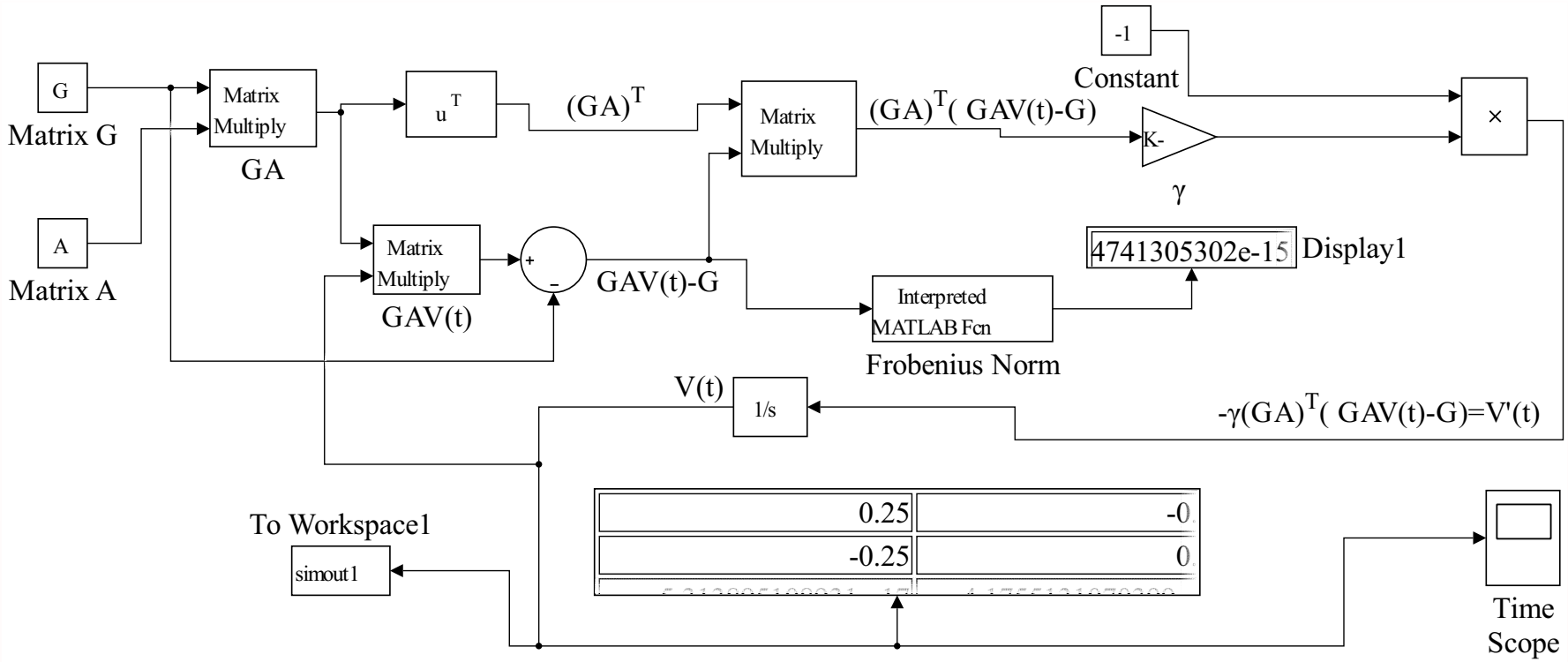


Figure 15: Simulink implementation of GNN(GA, I, G).

RNN models arising from GNN models

One specific model for computing $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$ was defined in
[NECO 2016][I. Živković, P.S. Stanimirović, Y. Wei, *Recurrent Neural Network for Computing Outer Inverses*, Neural Computation **28:5** (2016), 970–998]
by omitting the constant term $(GA)^{\mathrm{T}}$ in the $GNN(GA, I, G)$ model:

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = -\gamma(GA)^{\mathrm{T}}\mathcal{F}\left(GAV(t)-G\right),\ V(0)=V_0.$$

The resulting dynamics is shortly termed in [NECO 2016] as GNNATS2 model, and defined as

$$\dot{V}(t) = -\gamma\,\mathcal{F}\left(GAV(t) - G\right),\ V(0) = 0. \tag{29}$$

RNN models arising from GNN models

The Simulink implementation of GNNATS2 dynamics (29) is illustrated in Figure 16.

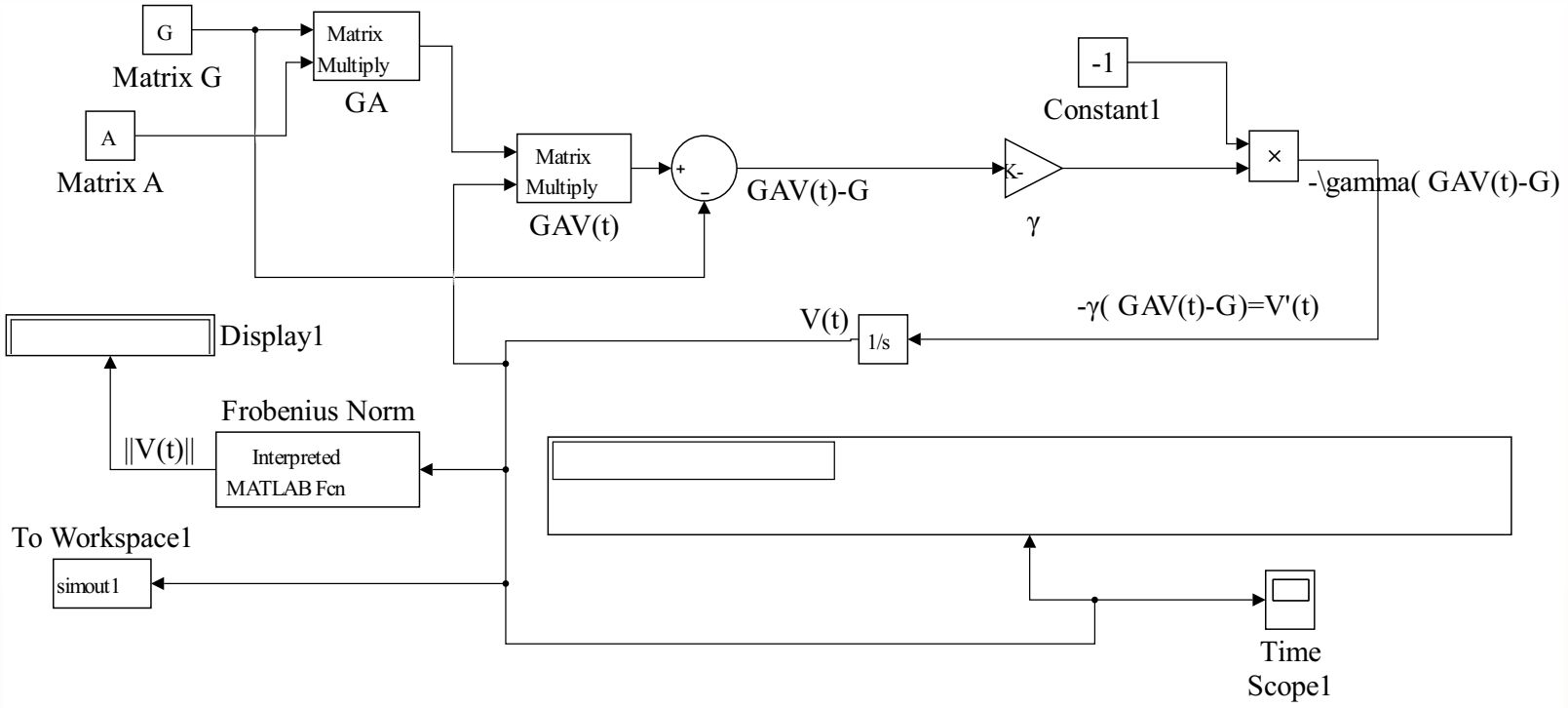


Figure 16: Simulink implementation of $\dot{V}(t) = -\gamma\,\mathcal{F}\,(GAV(t) - G)$.

RNN models arising from GNN models

Theorem 3

Let $A \in \mathbb{R}^{m\times n}$ be given matrix, $G \in \mathbb{R}_s^{n\times m}$ be arbitrary matrix satisfying $0 < s \leq r$, and

$$\sigma(GA) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$$

be the spectrum of GA. Suppose that the condition

$$\mathrm{Re}(\lambda_j) \geq 0,\ j = 1, 2, \ldots, n \tag{30}$$

is satisfied. Then:
(a) The exact solution to the RNN(GA, I, G) evolution $\frac{\mathrm{d}V_R(t)}{\mathrm{d}t} = -\gamma\left(GAV_R(t) - G\right)$ is equal to

$$V_R(t) = \mathrm{e}^{-\gamma t\, GA}V_R(0) + \left(I - \mathrm{e}^{-\gamma t\, GA}\right)A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}. \tag{31}$$

(b) If the initial approximation is the zero matrix: $V_R(0) = O$, then the exact solution to RNN(GA, I, G) is

$$V_R(t) = \left(I - \mathrm{e}^{-\gamma t\, GA}\right)A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}. \tag{32}$$

(c) the limiting value $\overline{V}$ of the GNNATS2 model produces the outer inverse $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$, i.e.,

$$\lim_{t\to\infty} V_R(0) = \overline{V}(0) = A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}. \tag{33}$$

RNN models arising from GNN models

Proof. (a) According to the linear dynamical systems theory [**?**], the closed-form solution of the state matrix $V(t)$ of RNN(GA, I, G) evolution is equal to

$$V_R(t) = \mathrm{e}^{-\gamma t\, GA} V_R(0) + \gamma \mathrm{e}^{-\gamma t\, GA} \int_0^t \mathrm{e}^{\gamma\, GA\, \tau} G \,\mathrm{d}\tau. \tag{34}$$

(c) Applying several elementary transformations and the basic property $GAA^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = G$ of the outer inverse $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$, it follows that

$$\begin{aligned}
\overline{V}_R(0) &= \lim_{t\to\infty} \gamma \mathrm{e}^{-\gamma GAt} \int_0^t \mathrm{e}^{\gamma GA\tau} G \,\mathrm{d}\tau \\
&= \left[\lim_{t\to\infty} \mathrm{e}^{-\gamma GAt} \int_0^t \mathrm{e}^{\gamma GA\tau} (\gamma GA) \,\mathrm{d}\tau\right] A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} \\
&= \left[\lim_{t\to\infty} \mathrm{e}^{-\gamma GAt} \mathrm{e}^{\gamma GA\tau} \Big|_{\tau=0}^{\tau=t}\right] A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} \\
&= \left[\lim_{t\to\infty} \mathrm{e}^{-\gamma GAt} \left(\mathrm{e}^{\gamma GAt} - I\right)\right] A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} \\
&= \left(I - \lim_{t\to\infty} \mathrm{e}^{-\gamma GAt}\right) A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}.
\end{aligned}$$

Then (33) can be verified using $\lim_{t\to\infty} \mathrm{e}^{-\gamma GAt} = O$. □

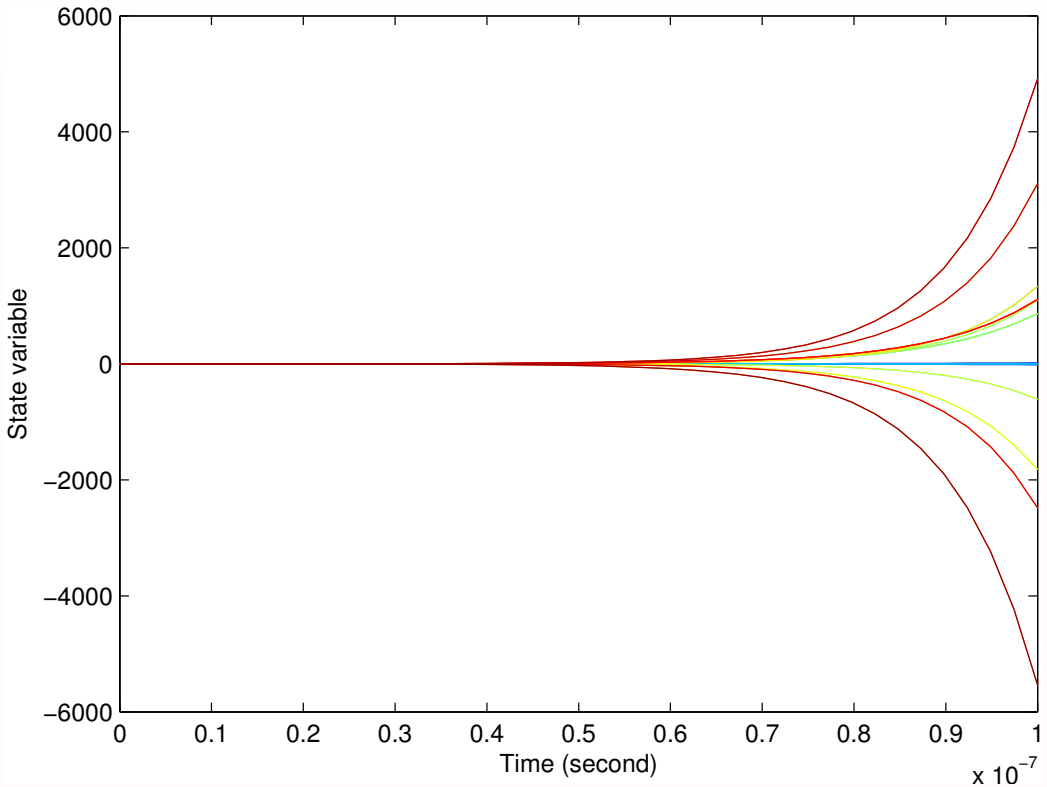
RNN models arising from GNN models

The application of the GNNATS2 model is conditioned by the properties of the spectrum of the matrix GA:

$$\sigma(GA) \subset \{z : \mathrm{Re}\,(z) \geq 0\}. \tag{35}$$

More precisely, the application of the GNNATS2 model diverges in the case when $\mathrm{Re}\,(\sigma(GA))$ contains negative values.

Consider
A=[1 -1 0 0 0 0; -1 1 0 0 0 0; -1 -1 1 -1 0 0; -1 -1 -1 1 0 0; -1 -1 -1 0 2 -1; -1 -1 0 -1 -1 2], G=-A'



RNN models arising from GNN models

An approach to assure the requirement (35):

$$\sigma(GA) \subset \{z : \mathrm{Re}\,(z) \geq 0\}$$

and recover global stability was proposed in
[I. Živković, P.S. Stanimirović, Y. Wei, *Recurrent Neural Network for Computing Outer Inverses*, Neural Computation **28:5** (2016), 970–998.]
and it is based on the replacement of G by $G_0 = G(GAG)^{\mathrm{T}}G$ in GNNATS2 model.

The leading idea is that G and G_0 produce the same outer inverse, because of $\mathcal{R}(G) = \mathcal{R}(G_0)$, $\mathcal{N}(G) = \mathcal{N}(G_0)$, and $\sigma(G_0A) \geq 0$.

But, this approach requires additional matrix multiplications during the computation of the matrix G_0 instead of G and sometimes causes numerical stability.

RNN models arising from GNN models

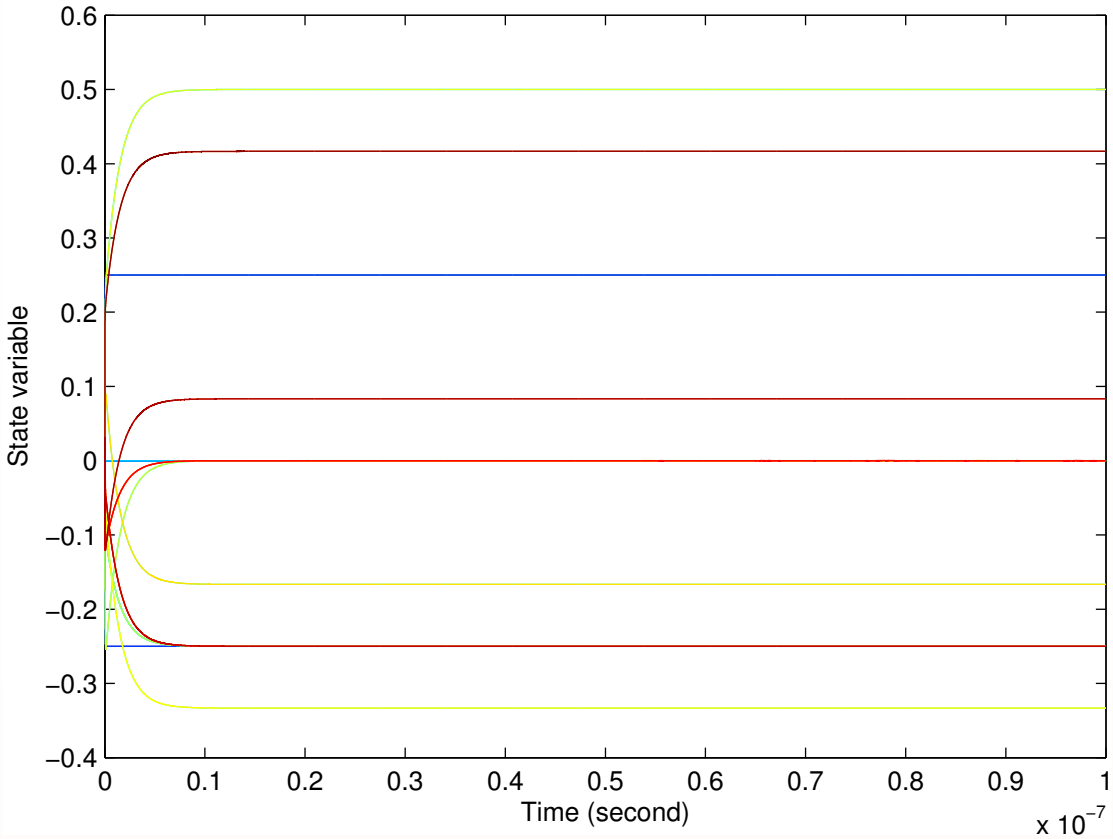


Figure 18: Convergence of the network when inappropriate G is replaced by G_0

RNN models arising from GNN models

A simplification of the GNN model which is applicable in computing the Drazin inverse A^{D} was proposed in
[IEEEE TNNLS 2015] P.S. Stanimirović, I. Živković, Y. Wei, Recurrent Neural Network for Computing the Drazin Inverse, IEEE Transactions on Neural Networks and Learning Systems, 26(11) (2015), 2830-2843.
This model can be derived removing the constant term $\left(A^{k+1}\right)^{\mathrm{T}}$ in GNN(A^k, A, A^k), $k \geq \mathrm{ind}(A)$. We can rewrite the equation (1^k) in the form

$$A^{k+1} V_D(t) - A^k = 0, \tag{36}$$

where $m \geq \mathrm{ind}(A)$, and $V_D \in \mathbb{R}^{n \times n}$ denotes the unknown matrix to be solved which corresponds to the matrix A^{D}. To solve for $V_D(t)$ via dynamic system approach, we can define a scalar-valued norm based error function:

$$\varepsilon_D(t) = \frac{\|A^{k+1} V_D(t) - A^k\|_F^2}{2}, \quad k \geq \mathrm{ind}(A).$$

Note that minimal value $\varepsilon(\bar{t}) = 0$ of the residual-error function $\varepsilon(t)$ is achieved in a minimum point $\overline{V} = V(\bar{t})$ if and only if $V(\bar{t})$ is the exact solution of $A^{k+1}X = A^k$.

RNN models arising from GNN models

The derivative of E with respect to $V \in \mathbb{R}^{n\times n}$ could simply be derived as

$$\frac{\partial \varepsilon_D(t)}{\partial V_D} = \left(A^{k+1}\right)^{\mathrm{T}} \left(A^{k+1} V_D(t) - A^k\right). \tag{37}$$

A particular RNN for computing the Drazin inverse A^{D} was proposed in [IEEEE TNNLS 2015]. This model can be derived removing the first constant term in GNN(A^k, A, A^k), $k \geq \mathrm{ind}(A)$, and it is defined as the following RNN(A^k, A, A^k) dynamics:

$$\dot{V}_D(t) = -\gamma \left(A^{k+1} V_D(t) - A^k\right), \quad k \geq \mathrm{ind}(A), \ V_D(0) = V_0. \tag{38}$$

Accordingly, an application of the model (38) is conditioned by

$$\mathrm{Re}\left(\lambda_j^{k+1}\right) \geq 0, \ j = 1, \ldots, n, \tag{39}$$

where $\sigma(A^{k+1}) = \{\lambda_1^{k+1}, \ldots, \lambda_n^{k+1}\}$ is the spectrum of A^{k+1} , $k \geq \mathrm{ind}(A)$ and $\{\lambda_1, \ldots, \lambda_n\}$ is the spectrum of A.
One method to resolve the limitation (39) is based on the possibility to find an appropriate power k such that (39) holds.
Another possibility to ensure the nonnegativity of the spectrum of the form (39) is based on the usage of the matrix $A^k \left(A^{2k+1}\right)^{\mathrm{T}} A^k$, $k = \mathrm{ind}(A)$.

Dynamical systems for computing DMP inverse

The GNN dynamics for computing the DMP inverse $A^{\mathrm{D},\dagger} = A^{\mathrm{D}}AA^{\dagger}$ is proposed in [DMP] [H. Ma, X. Gao, P.S. Stanimirović, Characterizations, iterative method, sign pattern and perturbation analysis for the DMP Inverse with its applications, Applied Mathematics and Computation 378 (2020), https://doi.org/10.1016/j.amc.2020.125196.]
The GNN for generating solutions to the matrix equation $GAX = G$ is given by the dynamical system

$$\dot{V}(t) = -\gamma\,(GA)^{\mathrm{T}}\,(GAV(t) - G)\,, \quad V(0) = V_0.$$

The authors of [**?**] defined the following simplified dynamical evolution for solving the matrix equation $GAX = G$:

$$\dot{V}(t) = -\gamma\,(GAV(t) - G)\,, \quad V(0) = O. \tag{40}$$

The dynamical evolution (40) will be termed as GNNATS2(GA, I, G).
The range space and null space of the DMP inverse are defined as follows:

$$\mathcal{R}(A^{\mathrm{D},\dagger}) = \mathcal{R}(A^k), \qquad \mathcal{N}(A^{\mathrm{D},\dagger}) = \mathcal{N}(A^kA^{\dagger}).$$

Dynamical systems for computing DMP inverse

Dynamical systems for computing DMP inverse Several additional properties of the DMP inverse are developed in Lemma 1 from [DMP] in order to define an appropriate error monitoring function $E(t)$ and corresponding GNN model.

Lemma 1

Let $A \in \mathbb{C}^{n\times n}$ satisfy $\mathrm{ind}(A) = k$. Then the DMP inverse $A^{\mathrm{D},\dagger}$ satisfies the following properties:

$$A^k A^\dagger A A^{\mathrm{D},\dagger} = A^k A^{\mathrm{D},\dagger} = A^k A^\dagger \tag{41}$$

$$A^{\mathrm{D},\dagger} A A^* = A^{\mathrm{D}} A A^* \tag{42}$$

$$A^k A^{\mathrm{D},\dagger} A = A^k \tag{43}$$

In view of (41), the dynamical system of the form (40) for computing $A^{\mathrm{D},\dagger}$ can be defined by dynamics GNNATS2($A^k A^\dagger A, I, A^k A^\dagger$), which produces

$$\dot{V}(t) = -\gamma\left(A^k A^\dagger A V(t) - A^k A^\dagger\right) = -\gamma\left(A^k V(t) - A^k A^\dagger\right), \quad V(0) = \mathbf{O}. \tag{44}$$

Dynamical systems for computing DMP inverse

Consider the matrix

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & -1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 0 & 2 & -1 \\ -1 & -1 & 0 & -1 & -1 & 2 \end{bmatrix} \tag{45}$$

satisfying $\mathrm{rank}(A) = 5$, $\mathrm{rank}(A^2) = \mathrm{rank}(A^3) = 4$. Therefore, $k = \mathrm{ind}(A) = 2$. The DMP inverse of A in the error-free form is given by

$$A^{\mathrm{D},\dagger} = A^2(A^5)^\dagger A^3 A^\dagger = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & -\frac{1}{4} & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & -\frac{5}{12} & -\frac{7}{12} & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & -\frac{7}{12} & -\frac{5}{12} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}. \tag{46}$$

Dynamical systems for computing DMP inverse

The condition $\mathrm{Re}(\lambda_j) \geq 0,\ j = 1, 2, \ldots, n$ is satisfied for A^2.
The architecture of GNNATS2($A^2, I, A^2A^\dagger$) is presented in the corresponding *Matlab* Simmulink implementation as in Figure 19.

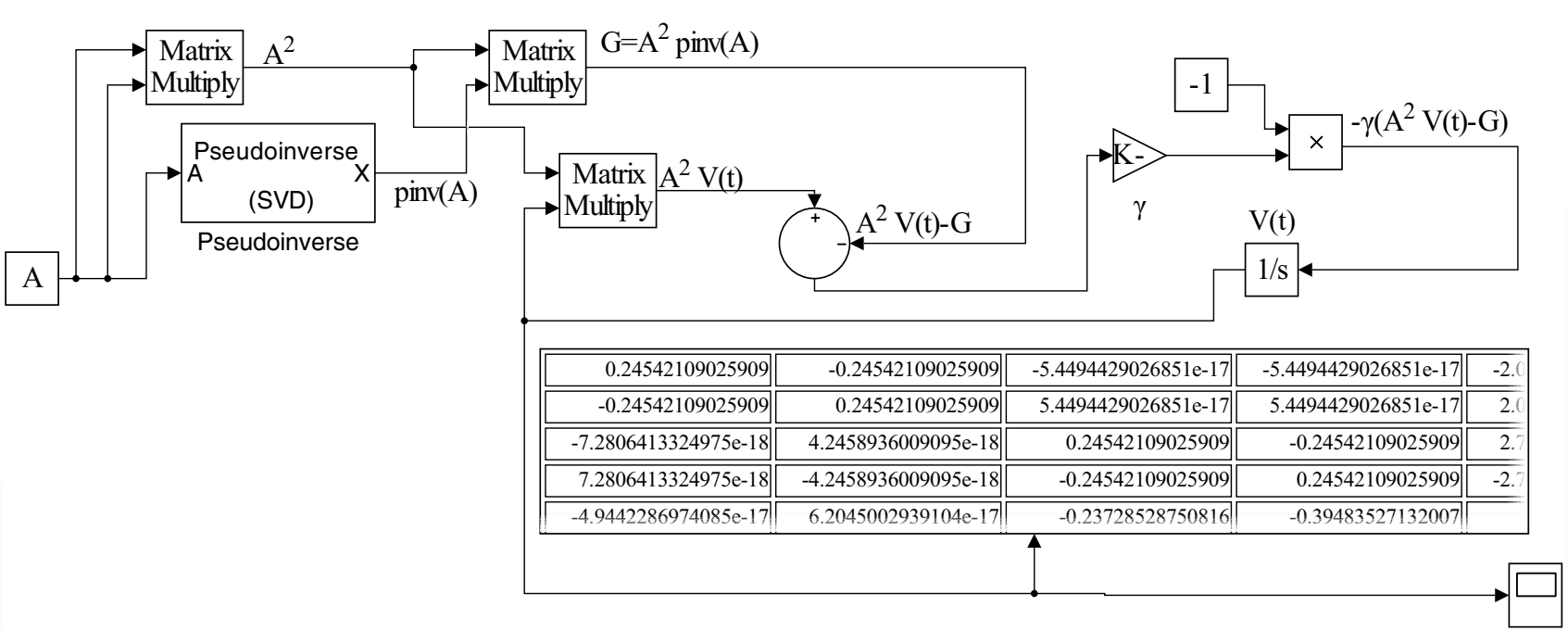


Figure 19: Simulink implementation of GNNATS2($A^2, I, A^2A^\dagger$).

Lecture 20:
- Existence and representations of solutions to some constrained matrix equations and systems of matrix equations.
- Computation of various generalized inverses arising from corresponding systems of matrix equations.

Outer generalized inverses and equations

Now, we consider algorithms based on the general framework consisting of two main steps:

1. Solve a particular linear matrix equation.
2. If necessary, multiply the solution derived in the first step by some appropriate matrices.

We consider several algorithms arising from this general framework whose results are certain inner or outer inverses with prescribed range and/or null space.
First results were published in
[Complexity 2017][P.S. Stanimirović, M. Ćirić, I. Stojanović, D. Gerontitis, *Conditions for existence, representations and computation of matrix generalized inverses*, Complexity, Volume 2017, Article ID 6429725, 27 pages.]

Theorem 4

Let $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{n\times k}$, $C \in \mathbb{C}^{l\times m}$.

(a) *The following statements are equivalent:*

(i) *there exists a $\{2\}$-inverse X of A satisfying $\mathcal{R}(X) = \mathcal{R}(B)$, denoted by $A^{(2)}_{\mathcal{R}(B),*}$;*

(ii) *there exists $U \in \mathbb{C}^{k\times m}$ such that $BUAB = B$;*

(iii) $\mathcal{N}(AB) = \mathcal{N}(B)$;

(iv) $B(AB)^{(1)}AB = B$, *for some (equivalently every)* $(AB)^{(1)} \in (AB)\{1\}$;

(v) $\operatorname{rank}(AB) = \operatorname{rank}(B)$.

(b) If the statements in **(a)** are true, then the set of all outer inverses with the prescribed range $\mathcal{R}(B)$ is represented by

$$\begin{aligned} A\{2\}_{\mathcal{R}(B),*} &= \left\{ B(AB)^{(1)} \mid \ (AB)^{(1)} \in (AB)\{1\} \right\} \\ &= \{BU \mid \ U \in \mathbb{C}^{k\times m}, \ BUAB = B\}. \end{aligned} \tag{47}$$

Moreover,

$$\begin{aligned} A\{2\}_{\mathcal{R}(B),*} &= \left\{ B(AB)^{(1)} + BY\left(I_m - AB(AB)^{(1)}\right) \,\middle|\, Y \in \mathbb{C}^{k\times m} \right\} \\ &= B(AB)\{1\}. \end{aligned}$$

Outer generalized inverses and equations

Theorem 4 provides not only criteria for the existence of an outer inverse $A^{(2)}_{\mathcal{R}(B),*}$ with prescribed range, but also provides a method for computing such an inverse.

Namely, the problem of computing a $\{2\}$-inverse X of A satisfying $\mathcal{R}(X) = \mathcal{R}(B)$ boils down to the problem of computing a solution to the matrix equation $BUAB = B$, where $U \in \mathbb{C}^{k\times m}$ is an unknown matrix.

If U is an arbitrary solution to this equation, then $X := BU$ is a $\{2\}$-inverse of A satisfying $\mathcal{R}(X) = \mathcal{R}(B)$.

Algorithm 1 Computing an outer inverse with prescribed range.

Require: Matrices $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times k}$.

1: Verify $\mathrm{rank}(AB) = \mathrm{rank}(B)$.
   If this condition is satisfied then continue.
2: Solve the matrix equation $BUAB = B$ with respect to $U \in \mathbb{C}^{k\times m}$.
3: Return $X = BU = A^{(2)}_{\mathcal{R}(B),*}$.

Outer generalized inverses and equations

Approach used in [Complexity 2017] for solving $BUAB = B$ is based on Gradient Neural Networks is based on the Gradient Neural Network (GNN) dynamical system for minimizing the Frobenius norm $\|\mathrm{BUAB{=}B}\|_F$.

The $GNN(B, AB, B)$ model for solving $BXAB = B$ is defined by

$$\dot{V}(t) = B^{\mathrm{T}} \mathcal{F}(B - BV(t)AB)(AB)^{\mathrm{T}} \tag{48}$$

It gives the solution $\tilde{V}_{V(0)} \in (AB)\{1\}$.
Then, according to Theorem 4

$$X := B\tilde{V}_{V(0)} \in A\{2\}_{\mathcal{R}(B),*}.$$

The Simulink implementation of Algorithm 1 in the set of real matrices is based on the G-GNN model. Since the goal is solving the matrix equation $B(t)U(t)A(t)B(t) = B(t)$, it is necessary to implement the model GNN(BU,B,B). *Matlab* Simulink implementation is presented in Figure 20.

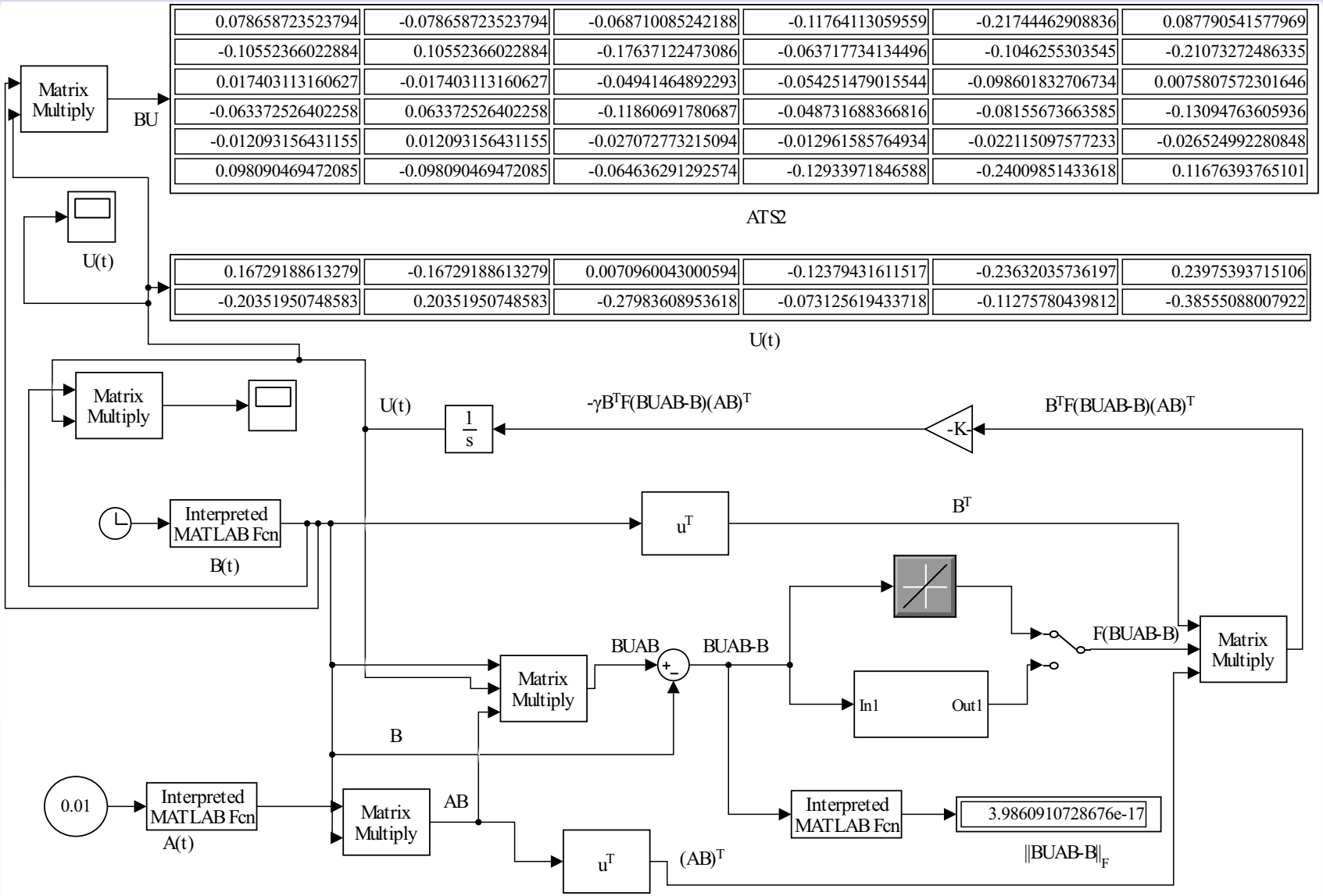


Figure 20: Simulink implementation of the GNN model for computing $BUAB = B$, $X = BU \in A\{2\}_{\mathcal{R}(B),*}$.

Outer generalized inverses and equations

The Simulink Scope and Display block denoted by $U(t)$ represents input signals corresponding to the solution $U(t)$ of the matrix equation $B(t)U(t)A(t)B(t) = B(t)$ with respect to the time t. The underlying G-GNN dynamics in the simulink presented in Figure 20 is

$$\dot{U}(t) = -\gamma B(t)^{\mathrm{T}} \mathcal{F}\left(B(t)U(t)A(t)B(t) - B(t)\right)\left(A(t)B(t)\right)^{\mathrm{T}}.$$

The Display block denoted by BU displays inputs signals corresponding to the solution $X(t) = B(t)U(t)$.

The block `Subsystem` implements the Power-sigmoid activation function and it is presented in Figure 21.

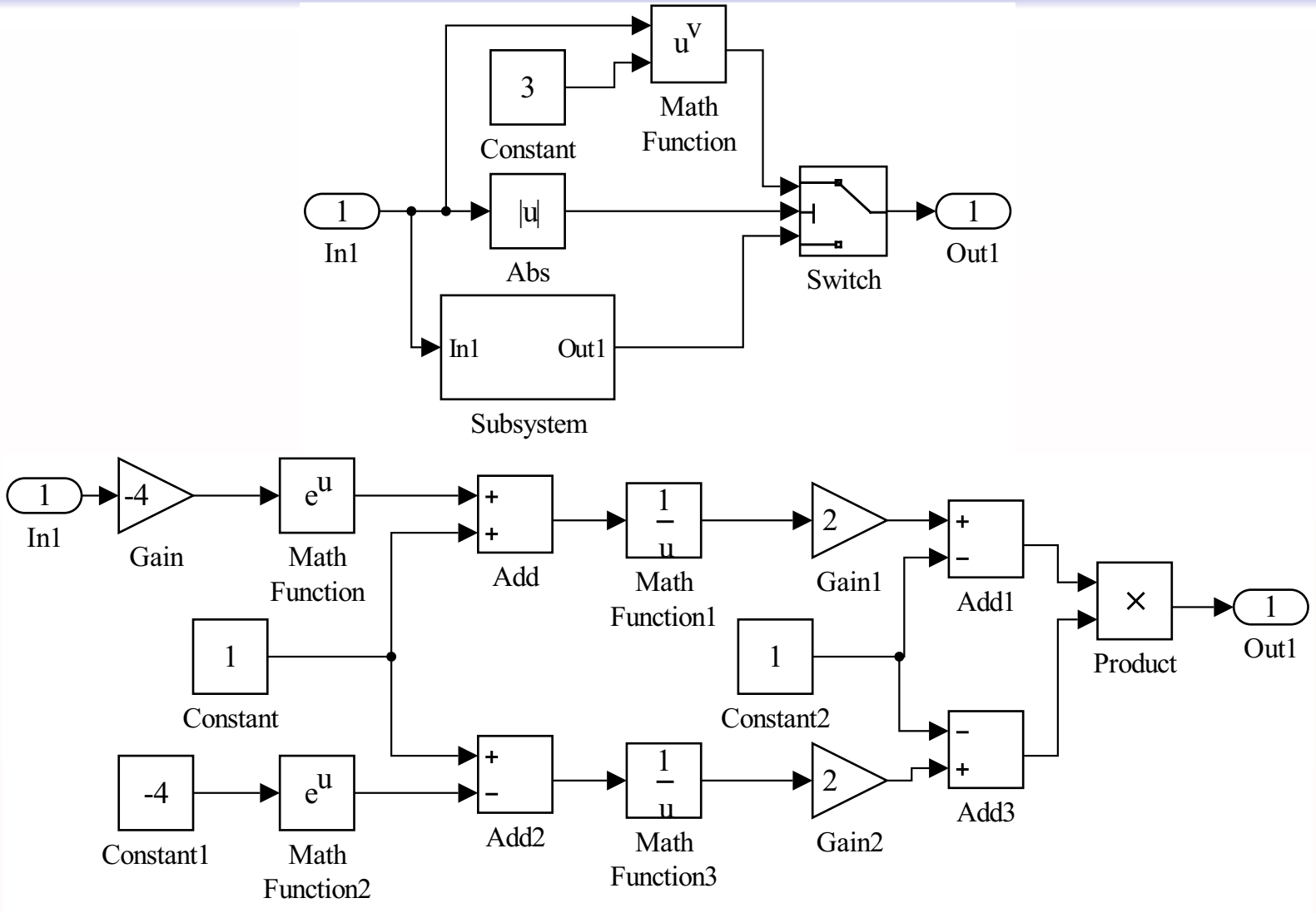


Figure 21: Block for the implementation of the Power-sigmoid activation function (left) and its Subsystem (right).

Output of RNN models in further calculations

Theorem 5

Let $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{n\times k}$ and $C \in \mathbb{C}^{l\times m}$.

(a) *The following statements are equivalent:*

(i) *there exists a $X \in A\{2\}$ satisfying $\mathcal{R}(X) = \mathcal{R}(B)$ and $\mathcal{N}(X) = \mathcal{N}(C)$;*

(ii) *there exist $U \in \mathbb{C}^{k\times l}$ such that $BUCAB = B$ and $CABUC = C$;*

(vi) $\mathcal{N}(CAB) = \mathcal{N}(B), \quad \mathcal{R}(CAB) = \mathcal{R}(C)$;

(vii) $\operatorname{rank}(CAB) = \operatorname{rank}(B) = \operatorname{rank}(C)$;

(viii) *$B(CAB)^{(1)}CAB = B$ and $CAB(CAB)^{(1)}C = C$, for some (equivalently every) $(CAB)^{(1)} \in (CAB)\{1\}$.*

(b) *If the statements in* **(a)** *are true, then the unique $\{2\}$-inverse of A with the prescribed range $\mathcal{R}(B)$ and null space $\mathcal{N}(C)$ is represented by*

$$\begin{aligned} A^{(2)}_{\mathcal{R}(B),\mathcal{N}(C)} &= B(CAB)^{(1)}C \\ &= BUC, \end{aligned}$$

for arbitrary $(CAB)^{(1)} \in (CAB)\{1\}$ and arbitrary $U \in \mathbb{C}^{k\times l}$ satisfying $BUCAB = B$ and $CABUC = C$.

Output of RNN models in further calculations

Algorithm for computing $A^{(2)}_{\mathcal{R}(B),\mathcal{N}(C)} = B(CAB)^{(1)}C$ was defined in [Complexity 2017]

Algorithm 2 Computing a $\{2\}$-inverse with the prescribed range and null space.

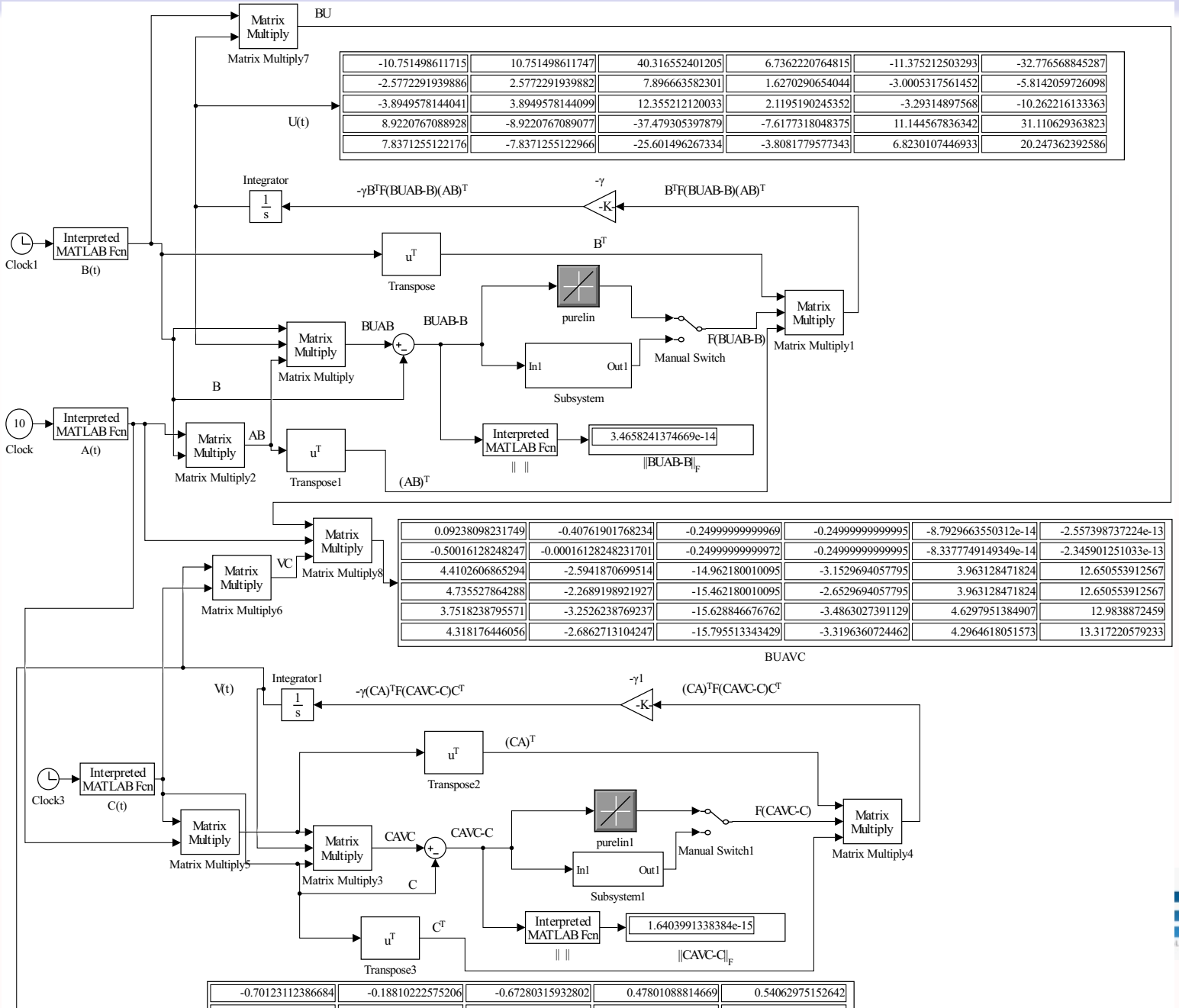
Require: Time varying matrices $A(t) \in \mathbb{C}^{m\times n}$, $B(t) \in \mathbb{C}^{n\times k}$ and $C(t) \in \mathbb{C}^{l\times m}$.

1: Verify $\mathrm{rank}(C(t)A(t)B(t)) = \mathrm{rank}(B(t)) = \mathrm{rank}(C(t))$.
If these conditions are satisfied then continue.

2: Solve the matrix equation $B(t)U(t)C(t)A(t)B(t) = B(t)$ with respect to an unknown matrix $U(t) \in \mathbb{C}^{k\times m}$.

3: Return $X(t) = B(t)U(t)C(t) = A(t)^{(2)}_{\mathcal{R}(B),\mathcal{N}(C)}$.

Output of RNN models in further calculations



-10.751498611715	10.751498611747	40.316552401205	6.7362220764815	-11.375212503293	-32.776568845287
-2.5772291939886	2.5772291939882	7.896663582301	1.6270290654044	-3.0005317561452	-5.8142059726098
-3.8949578144041	3.8949578144099	12.355212120033	2.1195190245352	-3.29314897568	-10.262216133363
8.9220767088928	-8.9220767089077	-37.479305397879	-7.6177318048375	11.144567836342	31.110629363823
7.8371255122176	-7.8371255122966	-25.601496267334	-3.8081779577343	6.8230107446933	20.247362392586

0.09238098231749	-0.40761901768234	-0.24999999999969	-0.24999999999995	-8.7929663550312e-14	-2.557398737224e-13
-0.50016128248247	-0.00016128248231701	-0.24999999999972	-0.24999999999995	-8.3377749149349e-14	-2.345901251033e-13
4.4102606865294	-2.5941870699514	-14.962180010095	-3.1529694057795	3.963128471824	12.650553912567
4.735527864288	-2.2689198921927	-15.462180010095	-2.6529694057795	3.963128471824	12.650553912567
3.7518238795571	-3.2526238769237	-15.628846676762	-3.4863027391129	4.6297951384907	12.9838872459
4.318176446056	-2.6862713104247	-15.795513343429	-3.3196360724462	4.2964618051573	13.317220579233

-0.70123112386684	-0.18810222575206	-0.67280315932802	0.47801088814669	0.54062975152642